
**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Povilas Mauručaitis**

# PROGRAMINĖS ĮRANGOS APSAUGA NUO NELEGALAUS NAUDOJIMO KEIČIANT JOS FUNKCIONALUMĄ

Baigiamasis magistro darbas

**Vadovas**
Doc. dr. Jonas Čeponis

**KAUNAS, 2016**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**
**KOMPIUTERIŲ KATEDRA**

# PROGRAMINĖS ĮRANGOS APSAUGA NUO NELEGALAUS NAUDOJIMO KEIČIANT JOS FUNKCIONALUMĄ

Baigiamasis magistro darbas

**Informacijos ir informacinių technologijų sauga (kodas 621E10003)**

**Vadovas**
(parašas) Doc. dr. Jonas Čeponis
(data)

**Recenzentas**
(parašas) Doc. dr. Romas Marcinkevičius
(data)

**Projektą atliko**
(parašas) Povilas Mauručaitis
(data)

**KAUNAS, 2016**

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos

(Fakultetas)

Povilas Mauručaitis

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga, 621E10003

(Studijų programos pavadinimas, kodas)

„Baigiamojo projekto pavadinimas“

**AKADEMINIO SĄŽININGUMO DEKLARACIJA**

20 16 m. Gegužės 23 d.

Kaunas

Patvirtinu, kad mano Povilo Mauručaičio baigiamasis projektas tema „Programinės įrangos apsauga nuo nelegalaus naudojimo keičiant jos funkcionalumą“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

*(vardą ir pavardę įrašyti ranka)* *(parašas)*

Mauručaitis, Povilas. Programinės įrangos apsauga nuo nelegalaus naudojimo keičiant jos funkcionalumą. Magistro baigiamasis projektas / vadovas doc. dr. Jonas Čeponis; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Mokslo kryptis ir sritis: 07T Informatikos inžinerija

Reikšminiai žodžiai: *programinės įrangos apsauga, nelegalus vartojimas, funkcionalumas, kriptografija.*

Kaunas, 2016. 46 p.

**SANTRAUKA**

*Šiuo metu programinės įrangos gamintojai naudoja įvairius licencijavimo ir apsaugos metodus, skirtus apsisaugoti nuo nelegalaus programų naudojimo. Šiame magistro baigiamajame darbe apžvelgiami egzistuojantys metodai skirti apsisaugoti nuo nelegalaus naudojimo. Taip pat pasiūlytas licencijavimo apsaugos patobulinimas, kurio esmė yra antrinis licencijavimas apie kurį vartotojas nežino. Šio antrinio licencijavimo metu aptikus nelegalaus panaudojimo faktą, šiek tiek pakeičiamas programinės įrangos funkcionalumas. Pokytis nėra itin akivaizdu, bet programos veikimas tampa nekorektiškas.*

Mauručaitis, Povilas. *Software Protection Against Unauthorized Use By Changing Its Functionality*: *Master's* thesis in Informatics Engineering supervisor assoc. prof. Jonas Čeponis. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: 07T Informatics Engineering

Key words: software protection, illegal use, functionality, cryptography

Kaunas, 2016. 46 p.

## SUMMARY

*Software manufacturers nowadays use various licensing and protection methods against illegal software use. An overview of existing protection from unauthorized software use methods presented in this Master's thesis. Licensing improvement suggested, which main point is secondary licensing. Secondary licensing is hidden from the end user. If the illegality fact is revealed during this secondary licensing, software functionality is being changed a little, in manner that it would not be obvious that it changed. But the calculations wouldn't be correct.*

**TURINYS**

## PAVEIKSLŲ SĄRAŠAS

## LENTELIŲ SĄRAŠAS

## TERMINŲ IR SANTRUMPŲ ŽODYNAS

PĮ – programinė įranga.

USB – universalioji jungtis (angl. *Universal Serial Bus*)

GUID – globaliai unikalus identifikatorius (angl. *Globally Unique Identifier*).

Emuliuoti – naudoti duomenų apdorojimo sistemą kitai duomenų apdorojimo sistemai imituoti tokiu būdu, kad būtų priimami tie patys duomenys, vykdomos tos pačios programos ir gaunami tie patys rezultatai, kaip ir tiesiogiai naudojant imituojamąją sistemą.

MAC – tinklo plokštės adresas (angl. *Media Access Control Address*).

BIOS – svarbiausia įvesties / išvesties sistema (angl. *Basic Input/Output System*).

Valdomosios programavimo kalbos – programavimo kalbos, kurių kodas nėra tiesiogiai transliuojamas į mašininį kodą (angl. *Managed programming languages*).

AES – blokinis simetrinis šifravimo algoritmas (angl. *Advanced Encryption Standard*).

DES – blokinis simetrinis šifravimo algoritmas (angl. *Data Encryption Standard*).

RSA – viešojo rakto kriptosistema.

JSON – atviro standarto formatas, perduodantis duomenų objektus sudarytus ir atributo ir reikšmės porų (angl. *JavaScript Object Notation*).

ms – milisekundė.

# ĮVADAS

Darbas priklauso Informacijos ir informacinių technologijų sauga (kodas 621E10003) studijų programai.

Nelegaliai naudojama programinė įranga yra svarbi problema Lietuvos ir viso pasaulio kontekste. Remiantis statistiniais duomenimis [1], nelegalios ar nulaužtos programinės įrangos naudojimas sudaro virš 40 proc. Todėl šio darbo metu buvo iškeltas tikslas patobulinti programinės įrangos apsaugą nuo nelegalaus naudojimo. Šiam tikslui įgyvendinti iškeliami šie uždaviniai:

1. Išanalizuoti esamus būdus naudojamus programinės įrangos licencijavimui.
2. Išanalizuoti metodus naudojamus apsisaugojimui nuo atvirkštinės inžinerijos.
3. Pasiūlyti patobulinimus apsaugai nuo atvirkštinės inžinerijos.
4. Nustatyti, kaip panaudoti antrinį licencijavimą, kad jis nebūtų lengvai aptinkamas.
5. Realizuoti programos prototipą, kuriame būtų realizuota patobulinta apsauga nuo atvirkštinės inžinerijos ir numatytas antrinis licencijavimas.

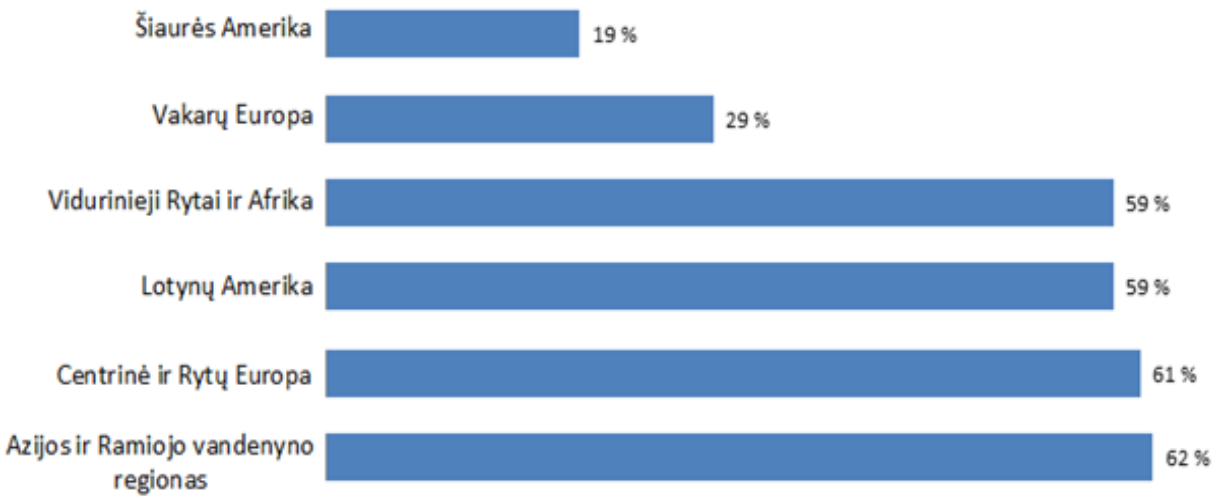
Darbo metu buvo sukurtas patobulintas licencijavimo algoritmas, su dviem licencijos patikros lygiais: atviru ir paslėptu. Šis licencijavimo algoritmas sukurtas remiantis jau egzistuojančiais licencijavimo būdais ir orientuotas apsaugoti programinė įrangą nuo statinių sutrikdymų.

Šis magistro baigiamasis darbas yra sudarytas iš 4 skyrių:

- Analizės skyrius. Iškeliamas problemos aktualumas, apžvelgiama kitų autorių literatūra, apibūdinanti programinės įrangos pažeidžiamumus ir taikomus metodus jiems išvengti.
- Projektavimo skyrius. Šiame skyriuje numatoma kas bus sukurta, aprašoma numatomos realizacijos struktūra.
- Prototipo realizavimas. Aprašomas sukurtas prototipas, kokios programinės priemonės panaudotos kuriant šį darbą, sukurtos programinės įrangos architektūra bei naudojami techniniai apsaugos sprendimai.
- Bandymo rezultatų skyriuje aprašoma realizuoto prototipo bandymas realiomis sąlygomis ir gauti rezultatai.

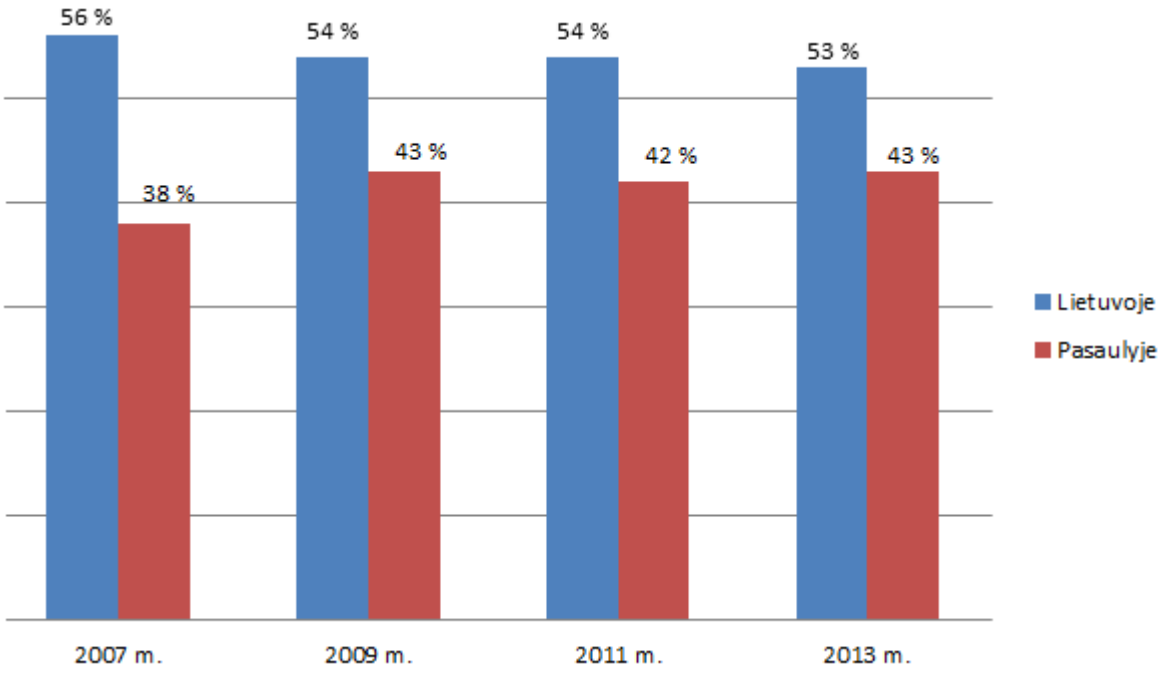
# 1. PROGRAMINĖS ĮRANGOS APSAUGOS NUO NELEGALAUS NAUDOJIMO KEIČIANT FUNCIONALUMĄ ANALIZĖ

Programinė įranga, toliau PĮ, yra intelektualinė nuosavybė. Nelegalus PĮ naudojimas, manoma, kad kenkia tiek programinę įrangą kuriančioms įmonėms – mažinant jų pelną, tiek ją perkantiems klientas – dėl išaugusios jos kainos [2]. Nelegalus programinės įrangos naudojimas – aktuali problema visame pasaulyje. Iš to uždirba žmonės nelegaliai platinantys programinę įrangą. Kadangi nelegali PĮ dažniausiai yra nemokama, todėl pelnas gaunamas iš įvairių portalų už reklamą ar paslapčia diegiant kenkėjiškas programas į vartotojų kompiuterius kartu su nelegalia PĮ [2]. 1 pav. matomi programinės įrangos piratavimo rodikliai pagal regionus pasaulyje 2013 metais.




**1 pav.** Programinės įrangos piratavimo rodikliai pagal regionus 2013 metais [1]

Lietuvoje nelegalios PĮ problema yra itin opi. Tai parodo 2 pav. pavaizduoti statistiniai duomenys apie piratinės PĮ naudojimą Lietuvoje.




**2 pav.** Piratinės programinės įrangos naudojimas Lietuvoje ir pasaulyje [1]

## 1.1 Sprendžiama problema

Programinės įrangos nelegalus naudojimas, panaudojus nelegalią priminę licenciją ar naudojant atvirkštinę inžineriją, randant saugomo mechanizmų pažeidžiamumus ir juos išnaudojant.

## 1.2 Darbo tikslas

Patobulinti programinės įrangos apsaugą nuo nelegalaus panaudojimo.

## 1.3 Darbo uždaviniai

Siekiant įgyvendinti darbo tikslą, bus sprendžiami tokie uždaviniai:

1. Išanalizuoti esamus būdus naudojamus programinės įrangos licencijavimui.
2. Išanalizuoti metodus naudojamus apsisaugojimui nuo atvirkštinės inžinerijos.
3. Pasiūlyti patobulinimus apsaugai nuo atvirkštinės inžinerijos.
4. Nustatyti, kaip panaudoti antrinį licencijavimą, kad jis nebūtų lengvai aptinkamas.
5. Realizuoti programos prototipą, kuriame būtų realizuota patobulinta apsauga nuo atvirkštinės inžinerijos ir numatytas antrinis licencijavimas.

Išsprendus visus aukščiau išvardintus uždavinius bus įgyvendintas darbo tikslas – patobulinta programinės įrangos apsauga nuo nelegalaus panaudojimo.

## 1.4 Apsauga nuo nelegalaus PĮ naudojimo

Siekiant užkirsti kelią nelegaliam PĮ disponavimui, į kurį įeina platinimas, pardavinėjimas ar naudojimas, naudojami įvairūs licencijavimo ir patikros metodai [3].

Vienas iš būdų apsaugoti licencijuotai programinei įrangai nuo kopijavimo yra surinkti informaciją apie kompiuterį, diegiant PĮ. Tai gali būti įvairi aparatinės ar programinės įrangos analizė. Remiantis šia analize suformuojami darbinės aplinkos parametrai, kurie išsaugojami standžiajame diske. Kai darbinės aplinkos parametrai yra surinkti, galima kiekvieną kartą paleidžiant programą juos sulyginti su esamais darbinės aplinkos parametrais. Tokiu atveju, kai sulyginus sistemos parametrus galima daryti išvada, kad tai nėra tas pats įrenginys, į kurį buvo įdiegta programa, galima stabdyti programos kodo vykdymą. Tačiau šis apsaugos mechanizmas gali būti sutrikdytas panaudojus vieną iš metodų, kurie gali padėti praleisti parametrų tikrinimą arba juos sufalsifikuoti. Šie metodai skirstomi į dvi grupes: [4]

- Statiniai – analizuojama programų išeities kodai;
- Dinaminiai – stebimas programos vykdymas derinimo režimu, mėginama keisti jos vykdymo eigą specialiomis apsaugos trikdymui skirtomis programomis.

Sekančiuose skyriuose bus plačiau apžvelgiami metodai skirti išvengti statinių ir dinaminių trikdymų.

### 1.4.1 Apsaugos metodai, siekiant išvengti statinių trikdymų.

Yra keturi apsaugos metodai skirti išvengti statinių sutrikdymų [6]:

- Kriptografiniai metodai.
- Susiejimo su identifikatoriumi metodas.
- Perėjimas ir dėklu pagrįsti metodai.
- Manipuliacijos programos kodu.

Taikant kriptografinius metodus diegiant programą surenkami etaloniniai duomenys bei parametrai apie sistemą. Tada jie taikant kriptografinius algoritmus yra užkoduojami į vieną iš sričių standžiajame diske. Išskiriami du pagrindiniai kriptografiniai metodai skirti apsaugai nuo kopijavimo:

Vienos krypties funkcijos. Jei surinktas aplinkos parametrų rinkinys susideda iš n parametrų, vienas elementas yra $x_i$ kai $i \in [1..n]$, tuomet lengvai galima apskaičiuoti $f(x_i) = y_i$, su kiekvienu i, tačiau $y_i = f(x_i)$ yra sąlyginai daug sunkiau apskaičiuoti su visomis i reikšmėmis. Taip užšifruotus ir standžiajame diske saugomus aplinkos parametrus praktiškai neįmanoma iššifruoti ir gauti aplinkos vykdymo parametrų.

Šifravimas raktu. Etaloniniai vykdymo aplinkos parametrai užšifruojami raktu, kuris ir yra tie patys etaloniniai parametrai. Paleidus programą, surinkti nauji duomenys užšifruojami etaloniniais duomenimis. Taikant šį metodą lyginami parametrai sutampa tik tuo atveju, jei etaloniniai ir esami sistemos parametrai sutampa.

Susiejimo su identifikatoriumi metodas. Jis naudojamas tada, kai programinės ir techninės kompiuterio įrangos parametrai negali būti pasiekti arba jų pasiekimas labai sulėtina programos paleidimą. Šio metodo veikimo principas yra toks, kad diegiant programą, formuojamas ir į standųjį diską įrašomas unikalus identifikatorius. Tada kiekvieną kartą leidžiant programą jis yra tikrinamas ir jei jis nerandamas arba nėra originalus, tuomet programa blokuoja savo kodo vykdymą.

Siekiant apsaugoti unikalų identifikatorių nuo kopijavimo, jis turi būti įrašytas į šias standžiojo disko vietas:

- Į duomenų srities klasterius, kurie failų atributų lentelėje pažymimi kaip užimti arba defektiniai.
- Į sistemos sričiai rezervuotus standžiojo disko sektorius.

Identifikatorius taip pat gali būti įkeltas į „USB“ atmintinę bei susietas su atmintuko unikaliu parametru. Tačiau šis apsaugos nuo kopijavimo metodas apsunkina vartotoją, kadangi kiekvieną kartą paleidžiant programinę įrangą vartotojui reikia prijungti ir atmintuką.

Perėjimais ir dėklu pagrįsti metodai numato įvairių dinamiškai besikeičiančių perėjimo ir pertraukčių komandų įtraukimą į programos kodą. Šios komandos gali būti generuojamos automatiškai.

Yra du pagrindiniai metodai, skirti manipuliuoti programos kodu:

- Tuščių modulių įtraukimas į programos kamieną. Įtraukiami papildomi programiniai moduliai, turintys daug funkcijų, kurios realiai programos vykdymo logikai nedaro jokios įtakos. Šie modulių neveiksnumas neturi būti akivaizdus įsilaužėliui.
- Apsaugotos programos pakeitimas. Pakeičiama programos pradžia, kad įsilaužėlis, naudodamas standartinį atvirkštinį asemblerio transliatorių, negalėtų teisingai išversti programos kodą į asemblerio kalbą. Šiam tikslui pasiekti naudojamos įvairios programos, pavyzdžiui: „Nota“ ar „Copylock“. Jos modifikuoja .exe failo antraštę, taip užkirsdamos kelią kopijavimui.

### 1.4.2 Apsaugos metodai siekiant išvengti dinaminio sutrikdymo

Siekiant apsaugoti programinę įrangą nuo dinaminio kopijavimo naudojami keletas metodų [6]:

- Programos vykdymo metu skaičiuojama jos užimamos atminties kontrolinės sumos – tai leidžia nustatyti vykdymo sekimą derintuvu.
- Neužimtos atminties palyginimas su įprastu programai – leidžia nustatyti ir apsisaugoti nuo veikos sekimo rezidentiniais įrankiais.
- Programai nepriskirtų atminties sričių turinio tikrinimas – leidžia programai pasiekti monopolinį darbo režimą.

- Pertraukimų vektorių turinio tikrinimas. Dažniausiai yra tikrinami 13h ir 21h vektoriai. Metodo esmė yra patikrinti, ar nėra neįprastų reikšmių.
- Pertraukimų vektorių atnaujinimas. Kopijuojamas pasirinktų vektorių turinys į laisvųjų vektorių sritį ir pakeičiami pertraukimų kreipiniai.
- Nuolatinis pertraukimų komandų uždraudimo ir leidimo komandų kaitaliojimas. Taip derintuvei sunkiau įterpti kontrolinius sekimo taškus.
- Nuolatinė programinės įrangos modulių vykdymo laiko kontrolė. Taip aptinkami programos sustojimai vykdomajame modulyje.

Išvardinti metodai leidžia nustatyti padarytus ar daromus programų pakeitimus, vykdymo eigos sekimą derintuve ir imtis priemonių, padėsiančių apsaugoti programinę įrangą.

### 1.4.3 Programinio kodo maskavimas

Programinio kodo maskavimas yra apsaugos metodas, kuris taikomas siekiant apsaugoti programos išeities kodą nuo neteisėto jo skaitymo. Maskuojant programos kodą jis yra pakeičiamas taip, kad lengvai skaitomi ir suprantami programos kintamųjų, funkcijų, klasių ar kitų objektų pavadinimai būtų pakeisti į žmogui sunkiai suprantamus įvairių simbolių sekas ar globalius unikalius identifikatorius – GUID [7].

Kuriant įvairius programinius sprendimus dažnai iškyla noras apsaugoti programinį kodą nuo nesankcionuotos prieigos, jo dalies ar viso nelegalaus kopijavimo ir per – panaudojimo [8]. Tai aktualu programoms, kurios kurtos naudojant valdomąsias programavimo kalbas ar kalbas, kurios nėra kompiliuojamos į mašininį kodą. Tai tokios kalbos, kaip „Java“, „C#“ ir kitos. Taip yra, nes turint sukompiliuotus šių kalbų bibliotekų failus, panaudojant nemokamai ar mokamai prieinamus dekompiliavimui skirtus įrankius, apžvelgiamus žemiau pateiktoje lentelėje, galima peržiūrėti žmogui lengvai perskaitomą programos kodą.

**1 lentelė.** Programos bibliotekų dekompiliavimui skirti įrankiai

| Įrankis | Skirtas programavimo kalbai | Kaina |
|---|---|---|
| „.NET Reflector“[9] | „C#“, „Visual Basic“ | Standartinė licencija – 73 EUR. Profesionali licencija – 155 EUR. Turi nemokamą bandomąją versiją. |
| „dotPeek“[10] | „C#“, „Visual Basic“ | Nemokamas |
| „DJ Java Decompiler“[11] | „Java“ | Nemokamas |
| „JD“ | „Java“ | Nemokamas |
| „JEB2“ | „Java“ | Standartinė licencija – 80 EUR/mėn. Verslui skirta licencija – 140 EUR/mėn., įmonėms skirta licencija – 280 EUR/mėn. Turi nemokamą bandomąją versiją. |

Iš 1 lentelės, kurioje apžvelgiami dekompiliavimui skirti įrankiai, matyti, kad šiuo metu populiariausiomis programavimo kalbomis sukurtoms bibliotekoms dekompiliuoti ir peržiūrėti jų turinį naudojami įrankiai yra lengvai prieinami internete ir dažnai nemokami.

Maskavimas dar svarbesnis yra programoms, parašytoms su „JavaScript“ kalba, kadangi tai yra kliento pusės programavimo kalba ir klientas gali be jokių papildomų įrankių peržiūrėti, kokius failus su programiniu kodu jam parsiuntė naršyklė.

**2 lentelė.** Programos išeities kodui maskuoti skirti įrankiai

| Įrankis | Skirtas programavimo kalbai | Kaina |
|---|---|---|
| „Zelix KlassMaster“ | „Java“ | Standartinė licencija – 460 EUR. Licencija skirta nedidelėms įmonėms – 210 EUR, įmonėms skirta licencija – 280 EUR/mėn. Turi nemokamą bandomąją versiją. |
| „Babel Obfuscator“[12] | „C#“, „Visual Basic“ | Standartinė licencija – 115 EUR. Profesionali licencija – 185 EUR. Įmonėms skirta licencija – 245 EUR. |
| ConfuserEx | „C#“, „Visual Basic“ | Nemokamas |
| Eazfuscator .NET | „C#“, „Visual Basic“ | Nemokamas |
| Yano Obfuscator | „C#“, „Visual Basic“ | Nemokamas |

Iš lentelės, skirtos maskavimui skirtiems įrankiams apžvelgti, matyti, kad įrankiai populiariausiomis programavimo kalboms obfuskuoti ir apsaugoti nuo nelegalaus skaitymo yra gana lengvai prieinami internete, tačiau mokami. Mažiau funkcionalumo turintys įrankiai – nemokamai.

## 1.5 Apgrąžos inžinerija

Bendru atveju apgrąžos inžinerija yra analizavimo procesas [13]. Dažnai šio proceso metu yra iškeliami du tikslai:

- Identifikuoti sistemos komponentus bei jų vidinius sąryšius.
- Atkurti ar perkurti sistemą į kitą formą ar didesnį abstrakcijos lygį [14].

Programinės įrangos apgrąžos inžinerija yra procesas, kurio metu išnaudojant programinį kodą kaip pagrindinį informacijos šaltinį, surenkama informacija apie funkcionuojančią sistemą [15].

## 1.6 Kriptografija

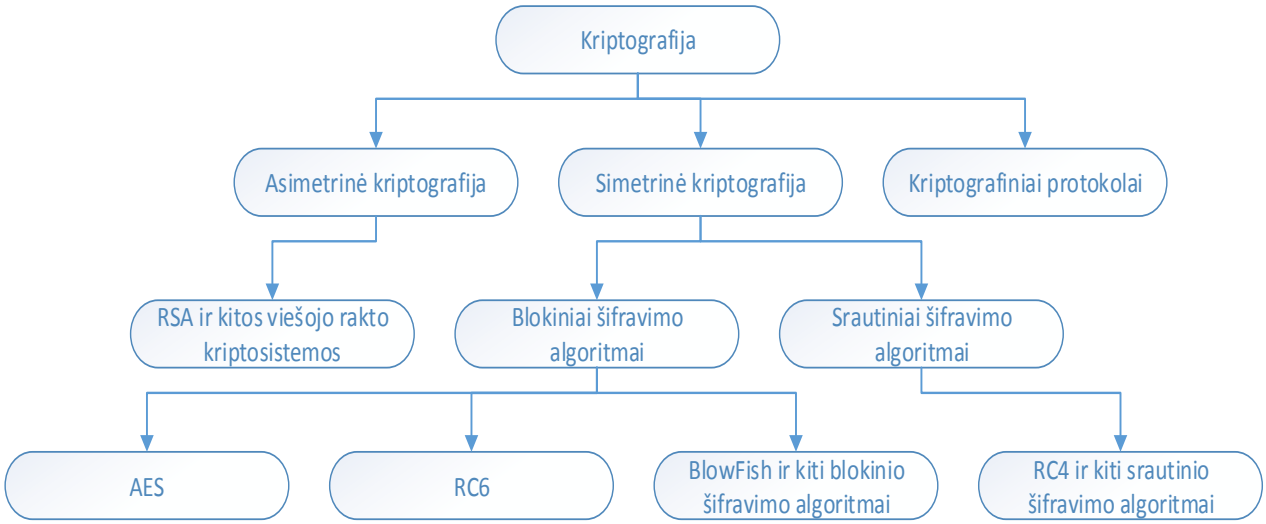
Remiantis „Kriptografijos teorija“ [16] knygos autoriais, šiuolaikinė kriptografija yra mokslo šaka, kuri sprendžia elektroninės informacijos saugos problemas.

Kriptografijos mokslui kyla keli svarbūs uždaviniai:

- Informacijos konfidencialumas (angl. *data confidentiality*). Užtikrinama, kad perduodamą slaptą informaciją galėtų perskaityti tik tas subjektas, kuriam ji skirta, o ne kažkas kitas, kam prieinamas kanalas, kuriuo perduodama informacija. Tai pasiekiama informaciją šifruojant.
- Informacijos vientisumas (angl. *data integrity*). Užtikrinama, kad siunčiamų duomenų niekas nesuklastotų. Dažniausiai tam pasitelkiamos maišos funkcijos (angl. *hash function*).
- Subjekto tapatumo nustatymas (angl. *subject authentification).* Užtikrinima, kad šifruotos informacijos siuntėjas tikrai yra tas subjektas, kuris teigia siuntęs informaciją, o ne apsimetėlis. Elektroninis parašas yra vienas iš autentifikavimo metodų.

Sujungus keletą išvardytų uždavinių gaunami dar du, nemažiau svarbūs:

- Subjekto anonimiškumas (angl. *subject anonimity)* – turi būti užtikrinama subjektų anonimiškumas.
- Informacijos autentiškumas (angl. *data authenticity)* – turi būti užtikrinama, kad duomenys yra teisingi ir jų autorius yra siuntėjas.




**3 pav.** Kriptografijos kryptys [17]

Kaip matyti iš aukščiau pateikto paveikslo (3 pav.) – išskiriamos trys pagrindinės kriptografijos kryptys:

- Simetrinė kriptografija. Kuri skirstoma dar į dvi kryptis: srautinius šifravimo algoritmus ir blokinius šifravimo algoritmus.
- Asimetrinė (viešojo rakto) kriptografija.
- Kriptografiniai protokolai.

Simetrinės ir asimetrinės kriptografijos kryptys bei hibridinės kriptosistemos privalumai plačiau aptariami kituose šio skyriaus poskyriuose.

### 1.6.1 Simetrinė kriptografija

Simetrinėje kriptografijoje duomenys užšifruojami ir iššifruojami tuo pačiu kriptografiniu raktu. Simetrinės kriptografijos greitis yra kur kas didesnis nei asimetrinės (vidutiniškai 100 kartų) bei patys raktai yra kur kas mažesni – 128, 256 bitų ilgio raktai laikomi saugiais [18].

Prieš perduodant duomenis tarp dviejų šalių, turi būti pasidalintas slaptasis raktas k naudojant vieną iš raktų apsikeitimo algoritmų, tokių kaip „Diffie-Hellman“ [19] ar kitu būdu per saugų kanalą. Duomenys užšifruojami naudojant šifravimo algoritmą E ir slaptąjį raktą k, taip gaunant šifrogamą c, c = E(k, m). Šifrograma siunčiama gavėjui, kuris gavęs pranešimą jame esančią informaciją iššifruoja naudodamas raktą k ir iššifravimo algoritmą D bei atstato pradinę žinutę m, m = D(k, c).

Simetrinėje kriptografijoje naudojami algoritmai skirstomi į srauto ir blokinius. Srautiniai algoritmai šifruoja kiekvieną duomenų simbolį $k_i \in K$, dažnu atveju šis simbolis yra dvejetainis skaičius ar baitas. Žinomiausias ir dažniausiai praktikoje taikomas srautinio šifravimo algoritmas „Rivest Cipher 4“.

Blokiniai algoritmai iš karto šifruoja fiksuoto dydžio duomenų bloką. Dažnu atveju duomenų bloko dydis yra 64 („DES“) ar 128 („AES“) bitai.[20]

### 1.6.2 Asimetrinė kriptografija

Asimetrinės kriptografijos arba kitaip viešojo rakto kriptografijos pagrindas yra tai, kad kiekvieno subjekto kriptografinis raktas yra padalintas į dvi dalis: viešąjį raktą ir privatųjį raktą. Viešasis raktas yra visiems atvirai prieinamas ir jis skirtas duomenims užšifruoti. Tuo tarpu privatusis raktas žinomas tik pačiam subjektui ir jis skirtas duomenims iššifruoti. Asimetrinės kriptografijos raktų dydžiai turi būti pakankamai dideli. Vienos iš pirmųjų asimetrinės kriptografijos sistemos „RSA“ šiuo metu priimtas saugus rakto dydis privalo būti didesnis ar lygus 2048 bitams [21].

Dėl santykinai didelio savo rakto dydžio, viešojo rakto kriptografija yra apie 100 kartų lėtesnė už simetrinę kriptografiją.

Tarkime, kad turime du subjektus: duomenų siuntėją ir gavėją. Gavėjas turi savo raktų porą: privatųjį raktą sk ir viešąjį raktą pk. Siuntėjas siųsdamas duomenis m juos užšifruoja viešuoju gavėjo raktu naudodamas šifravimo funkciją E ir gauna šifrogramą c, c = E(pk, m). Tuomet gavėjas gali iššifruoti gautą šifrogramą naudodamas savo privatųjį raktą sk ir atstatyti pradinius duomenis m.

Keletas svarbiausių viešojo rakto kriptografinių sistemų yra „RSA“, „ELGamalio“ ir „Rabino“. Visos šios kriptosistemos suteikia galimybę šifruoti bei pasirašyti skaitmeniniu parašu.

### 1.6.3 Hibridinė kriptosistema

Hibridinė kriptosistema yra tokia sistema, kurioje naudojama abi kriptografijos šakos – simetrinė ir asimetrinė. Šios kriptosistemos naudojamos perduoti dideliems duomenų kiekiams nenaudojant raktų apsikeitimo protokolų. Šios kriptosistemos esmė yra ta, kad duomenys perduodami užšifruoti simetriniu raktu. Taip jie užšifruojami bei iššifruojami greičiau, todėl sunaudojama mažiau resursų, o pats simetrinis raktas gavėjui nusiunčiamas panaudojus viešojo rakto kriptografiją. Simetrinis raktas užšifruojamas viešuoju asimetriniu gavėjo raktu ir jam išsiunčiamas. Tuomet gavėjas gauna raktą, kuris buvo jam saugiai atsiųstas tinklu, nenaudojant papildomų protokolų.

Naudojant hibridinę kriptosistemą išnaudojama patogus simetrinis šifravimas apsikeitimui raktais ir greitas simetrinis šifravimas.

## 1.7 Išvados

Nelegaliai naudojama programinė įranga yra itin svarbi problema šiandieninėje Lietuvos bei tarptautinėje visuomenėje. Remiantis statistiniais 2007 – 2013 m. duomenimis nelegalios ar nulaužtos programinės įrangos naudojimas sudaro 38 – 43 proc. Tai yra didelis skaičius. Programinės įrangos verslu užsiimančioms įmonėms kasmet patiriant didelius nuostolius dėl nelegalaus jų kuriamo produkto panaudojimo, todėl svarbu sukurti kuo sunkiau nulaužiamus licencijavimo metodus.

Kuriant saugius licencijavimo būdus naudojami įvairūs metodai. Šie metodai apima matematinius modelius, su operacinių sistemų veikimu susijusius modelius ir kitus metodus, kurie šiame darbe nėra plačiau nagrinėjami. Dažnai siekiant apsaugoti programinę įrangą nuo įvairių sukrikdymų taikomi kriptografiniai – simetriniai, asimetriniai šifrai, išeities kodo maskavimo algoritmai ar net fiziniai įrenginiai, kuriuose vykdoma dalis funkcionalumo.

Atlikus analizę jau egzistuojančių licencijavimo metodų bus sukurtas patobulintas licencijavimo modelis. Jis apims dviejų lygių licencijos patikros mechanizmą, kur pirminis licencijavimas bus atviras ir matomas vartotojui, o antrinis licencijavimo procesas kiek įmanoma paslėptas ir sunkiai aptinkamas.

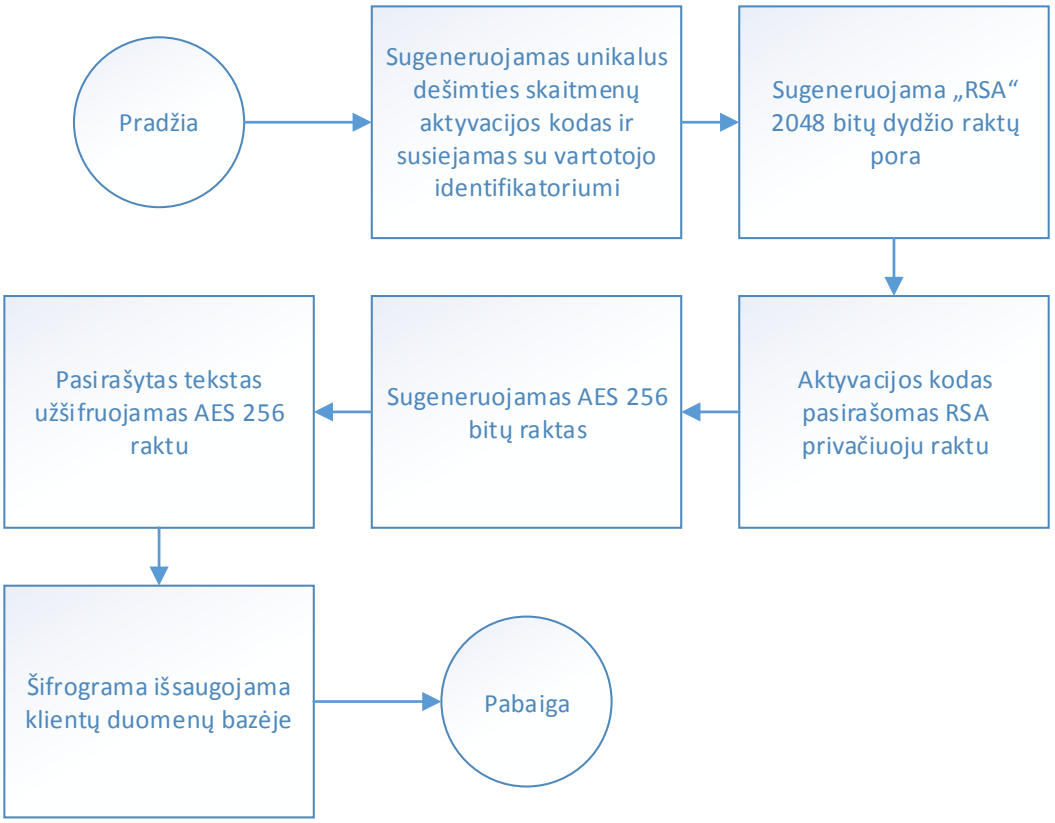
# 2. PROGRAMINĖS ĮRANGOS APSAUGOS NUO NELEGALAUS NAUDOJIMO KEIČIANT FUNCIONALUMĄ PROJEKTAS

Šiuo metu sukurta gausi įvairovė metodų, skirtų apsaugoti programinę įrangą nuo nelegalaus panaudojimo, tačiau dažniausiai praktikoje yra taikomas aktyvacijos kodo reikalavimas. Visgi dar nėra sprendimo, kuris būtų visiškai patikimas ir apsaugotų programinę įrangą nuo nelegalaus panaudojimo. Todėl šio darbo metu bus sukurtas sprendimas, kuris padės aptikti faktą, kad programinė įranga yra naudojama nelegaliai. Šis sprendimas apims atsitiktiniu laiko momentu iškviečiamą antrinę licencijos validumo patikrinimo funkciją, kuri remdamasi tam tikrais iš anksto numatytais reikalavimais galės aptikti, kad programinė įranga veikia nelegaliai.

## 2.1 Licencijos sudarymo procesas

Programinės įrangos kūrėjas, siekiantis užtikrinti, kad jo programinė įranga nebūtų naudojama nelegaliai, privalo realizuoti pakankamai patikimą licencijavimo metodą. Šiame poskyryje aprašomas licencijos generavimo procesas.

Kūrėjas licencijos sudarymo metu turi sugeneruoti dešimties skaitmenų aktyvacijos kodą, kuris susiejamas su programinės įrangos pirkėjo identifikatoriumi, taip užtikrinama, kad aptikus nelegaliai naudojamą programą būtų galima sužinoti, su kurio kliento aktyvacijos kodu ji naudojama. Taip pat sugeneruojama „RSA“ asimetrinių viešojo ir privačiojo raktų pora, kurių viešasis raktas įrašomas į dinaminę biblioteką ir taip pat keliaus kartu su programa, o privatusis liks pardavėjui, su juo pasirašomas aktyvacijos kodas ir pasirašytas tekstas įrašomas į licencijos failą. Tuomet bus sugeneruotas „AES“ simetrinės kriptografijos raktas, kuriuo bus užšifruojama licencija bei pats raktas keliaus su programa.



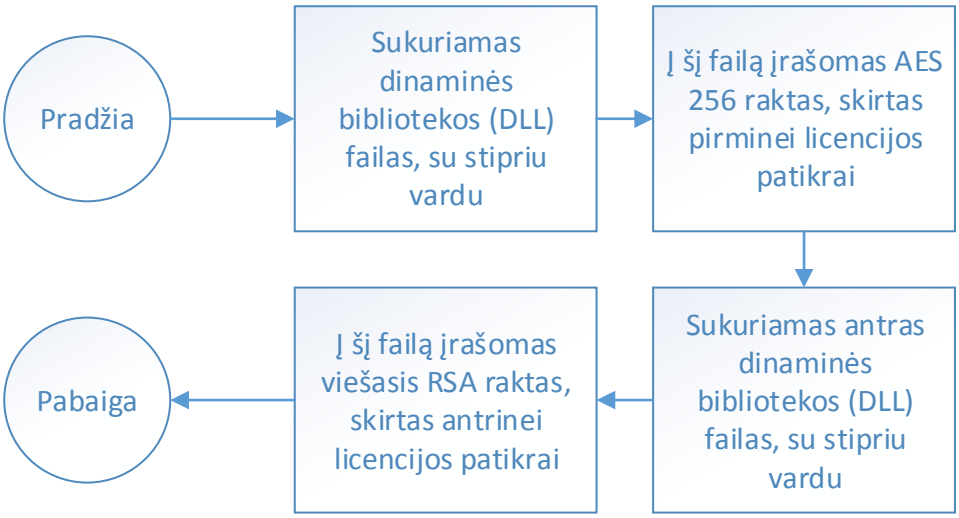

**4 pav.** Licencijos sudarymo proceso diagrama

Tai yra pradinė pasiruošimo licencijavimui fazė, kurios metu sugeneruojami kriptografiniai raktai bei licencija.

## 2.2 Dinaminių bibliotekų paruošimo procesas

Dar vienas licencijavimo proceso parengimo etapas yra dinaminių bibliotekų, kurios bus susietos su atskira programos instancija paruošimas. Turi būti parengtos programos bibliotekos, kurios bus skirtingos kiekvienai eksploatuojamai programinės įrangos instancijai.

Paruošiami du dinaminių bibliotekų (angl. *dll – dynamic-link library)* failai su stipriu vardu, taip užtikrinant, kad jie nebus pakeisti netikrais, vartotojo sukurtais failais. Į vieną iš jų įrašomas programai skirtas „AES“ kriptografinis raktas, kuris bus skirtas iššifruoti licencijos failui, o į kitą – įrašomas viešasis „RSA“ raktas, kuris bus naudojamas patikrinti programinės įrangos kūrėjo parašu pasirašyto aktyvacijos kodo autentiškumą.




**5 pav.** Programos bibliotekų paruošimo proceso diagrama

Programinės įrangos kūrėjui šis paruošimas būtų daugiausiai kaštų reikalaujantis darbas, kadangi kiekvienai instancijai turi būti paruošti dinaminių bibliotekų failai ir keliauti kartu su programos instaliacijos failu bei šių bibliotekų turinys turi būti unikalus kiekvienai aplikacijai. Šis procesas galėtų būti automatizuojamas panaudojant vieną iš programinės įrangos automatizuotų kompiliatorių. Tokių kaip „TFBuild“, „Jenkins.NET“ ar rašant įvairius skriptus modifikuojančius programinius failus, pakeičiant tam tikras reikšmes, kaip, kad raktai ar versijos ir naudojant „MSBuild“ įrankius paruošti programinės įrangos versijas. kurios bus siunčiamos klientams. Šiame darbe kuriamame prototipe šis procesas nebus automatizuotas, o atliekamas rankiniu būdu.

## 2.3 Pirminio licencijavimo procesas

Pirminis licencijavimo procesas yra visiškai atviras ir turi būti pakankamai sudėtingas, kad žmonės siekiantys jį nulaužti negalėtų numatyti fakto, jog naudojamas antrinis licencijavimas. Jei vis dėlto pirminis licencijavimas programišiams pasirodytų gana nesudėtingas, jie pradėtų gilintis toliau ir ieškoti paslėpto licencijos patikros metodo.

Pirminio licencijavimo procesas sudarytas taip, kad jis apsaugotų nuo aktyvacijos kodų generatoriaus, kadangi aktyvacijos kodai tikrinami kūrėjo serveryje. Jei pavyktų nukreipti užklausą siųstą kūrėjo serveriams į suklastotus serverius, taip mėginant apeiti pirminę patikrą, nebūtų gautas tinkamas programinės įrangos licencijos atsakas, jo nepavyktų iššifruoti ir nepavyktų nelegaliai paleisti programinės įrangos.

### 2.3.1 Aplinkos parametrų reikšmės

Diegiant programinę įrangą surenkami aplinkos parametrai ir išsaugoma jų maišos (angl. *hash)* reikšmės. Šie parametrai bus naudojami kiekvieną kartą paleidus programinę įrangą. Diegimo metu gali būti surinktos įvairios parametrų kombinacijos, pateiktos žemiau esančioje

lentelėje. Pasikeitimo tikimybė nurodo, kokia yra tikimybė, kad galėtų pasikeisti nurodytas parametras viename kompiuteryje. Pasikeitus keletui svarbių parametrų galima daryti prielaidą, kad pasikeitė ne tik tam tikri kompiuterio komponentai, o kad programa veikia kitame, nei ji buvo įrašyta fiziniame ar virtualiame įrenginyje. Jei parametro pasikeitimo tikimybė yra žema, tai reiškia, kad tikimybė tam elementui pasikeisti yra mažesnė nei 1 proc., vidutinė – 1–15 proc., o aukšta daugiau nei 15 proc. Parametro svarba nurodo, ar atitinkamam parametrui pasikeitus, licencijavimo algoritmas galėtų traktuoti, kokia yra tikimybė, kad pasikeitė programos veikimo aplinka.

**3 lentelė.** Galimi aplinkos parametrai

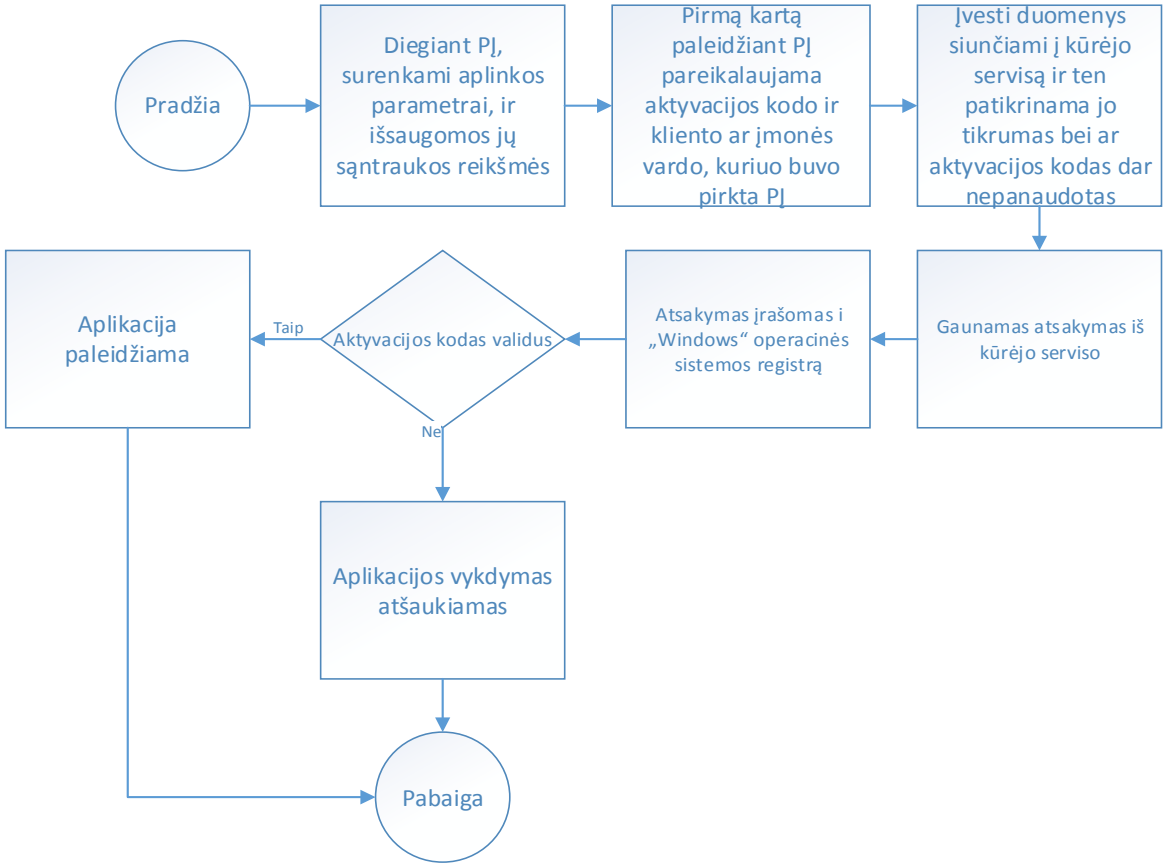
| Parametras | Pasikeitimo tikimybė | Svarba |
|---|---|---|
| Procesoriaus identifikatorius | Žema | Aukšta |
| Procesoriaus taktinis dažnis | Vidutinė | Žema |
| Procesoriaus architektūros tipas | Žema | Aukšta |
| Procesoriaus modelis | Žema | Aukšta |
| Pagrindinės plokštės identifikatorius | Žema | Aukšta |
| Sisteminio kietojo disko serijos numeris | Aukšta | Žema |
| Sisteminio kietojo disko modelis | Vidutinė | Vidutinė |
| Tinklo plokštės MAC adresas | Žema | Aukšta |
| Operatyviosios atminties kiekis | Aukšta | Žema |
| Operatyviosios atminties dėtuvių kiekis | Žema | Aukšta |
| Kompiuterio vardas | Žema | Vidutinė |
| BIOS – Serijos numeris | Žema | Aukšta |
| BIOS – Gamintojas | Žema | Aukšta |
| Operacinės sistemos versija | Vidutinė | Žema |
| Operacinės sistemos architektūros tipas | Žema | Aukšta |
| Vaizdo plokštės atminties kiekis | Vidutinė | Žema |
| Vaizdo plokštės serijos numeris | Vidutinė | Žema |

Galima surinkti ir daugiau aplinkos parametrų reikšmių, kurios šioje lentelėje nepateiktos. Visi įmanomi parametrai, kurie galėtų apibūdinti fizinę įrangą nėra pateikti, nes jų pasikeitimo tikimybė yra per didelė. Taip pat parametrai gali neturėti arba turėti labai mažą įtaką, kuri apibūdina kriterijų, pagal kurį būtų galima nuspręsti ar programinė įranga veikia kitame įrenginyje, nei kad buvo įdiegta.

Siekiant išvengti didelės tikimybės, kad aplinkos parametrai pasikeis tame pačiame kompiuteryje ir programinė įranga nebeveiks, nors įdiegta legaliai, būtina nustatyti ganėtinai aukštą pereinamąją ribą, kurią viršijus būtų stabdomas korektiškas programos veikimas.

## 2.4 Pirminio licencijavimo inicijavimas

Naudotojui, pirmą kartą paleidus programą, prašoma suvesti aktyvacijos kodą bei įmonės ar kliento kodą, kuris buvo nurodytas perkant programinę įrangą. Suvestas aktyvacijos kodas užšifruojamas viešuoju RSA raktu o kliento kodas atvira forma siunčiami į programinės įrangos kūrėjo serverį. Jame tikrinamas jų autentiškumas: ar toks klientas pirko programinę įrangą su tokiu licencijos raktu bei ar licencija dar nebuvo panaudota. Patikrinus duomenis ir įsitikinus, kad jie yra teisingi, serveris išduoda licencijos atsaką, kuriame yra licencijos kodas pasirašytas kūrėjo privačiu RSA raktu bei užšifruotas simetriniu „AES“ aplikacijos raktu. Priešingu atveju grąžinama klaidos žinutė. Gauti duomenys išsaugomi „Windows“ operacinės sistemos registre, kuriame saugoma kita programinės įrangos konfigūracija. Duomenys saugomi šifruotame pavidale, taip apsunkinant jų reikšmės gavimą programišiams. Sėkmingai įvykdžius šį licencijavimo procesą aplikacija paleidžiama vykdymui. Kitu atveju, jei aktyvacijos kodas neteisingas, kūrėjo serveris grąžina klaidą. Ši klaida taip pat yra užšifruojama simetriniu AES aplikacijos raktu bei įrašoma į registrą, taip apsunkinant prieigą prie šio klaidos pranešimo nurodančio, kad aktyvacijos kodas neteisingas. Taip programišiams būtų sunkiau, surasti skirtumą tarp teigiamo ir neigiamo serverio atsako bei suimituoti teigiamą atsaką. Įrašius į registrą aplikacijos paleidimo klaidos pranešimą aplikacijos vykdymas sustabdomas.




**6 pav.** Aktyvacijos kodo patikros proceso diagrama

Diegimo metu surinkti aplinkos parametrai yra tikrinami kiekvieną kartą paleidus programinę įrangą. Esminis reikalavimas, kad diegimo ir antrinio licencijavimo metu surinkti parametrai skirtųsi. Tokiu būdu nors ir būtų aptiktas pirminis parametrų rinkimas ir jų reikšmės būtų emuliuojamos (tai yra būtų imituojamos tokios aplinkos parametrų reikšmės kokios buvo diegiant programinę įrangą), tačiau antrinio licencijavimo metu naudojamos aplinkos parametrų reikšmės skirtųsi ir jos, ko gero, nebūtų emuliuojamos.

Pirminis licencijos patikros procesas yra sąlyginai saugus, tačiau didelę patirtį turintiems programišiams jį pavyktų nulaužti. Tad siekiant apsaugoti programinę įrangą nuo nelegalaus panaudojimo – taikomas antrinis licencijavimas, kurio detalesnis veikimo procesas aptariamas sekančiuose šio darbo poskyriuose.

## 2.5 Antrinio licencijavimo procesas

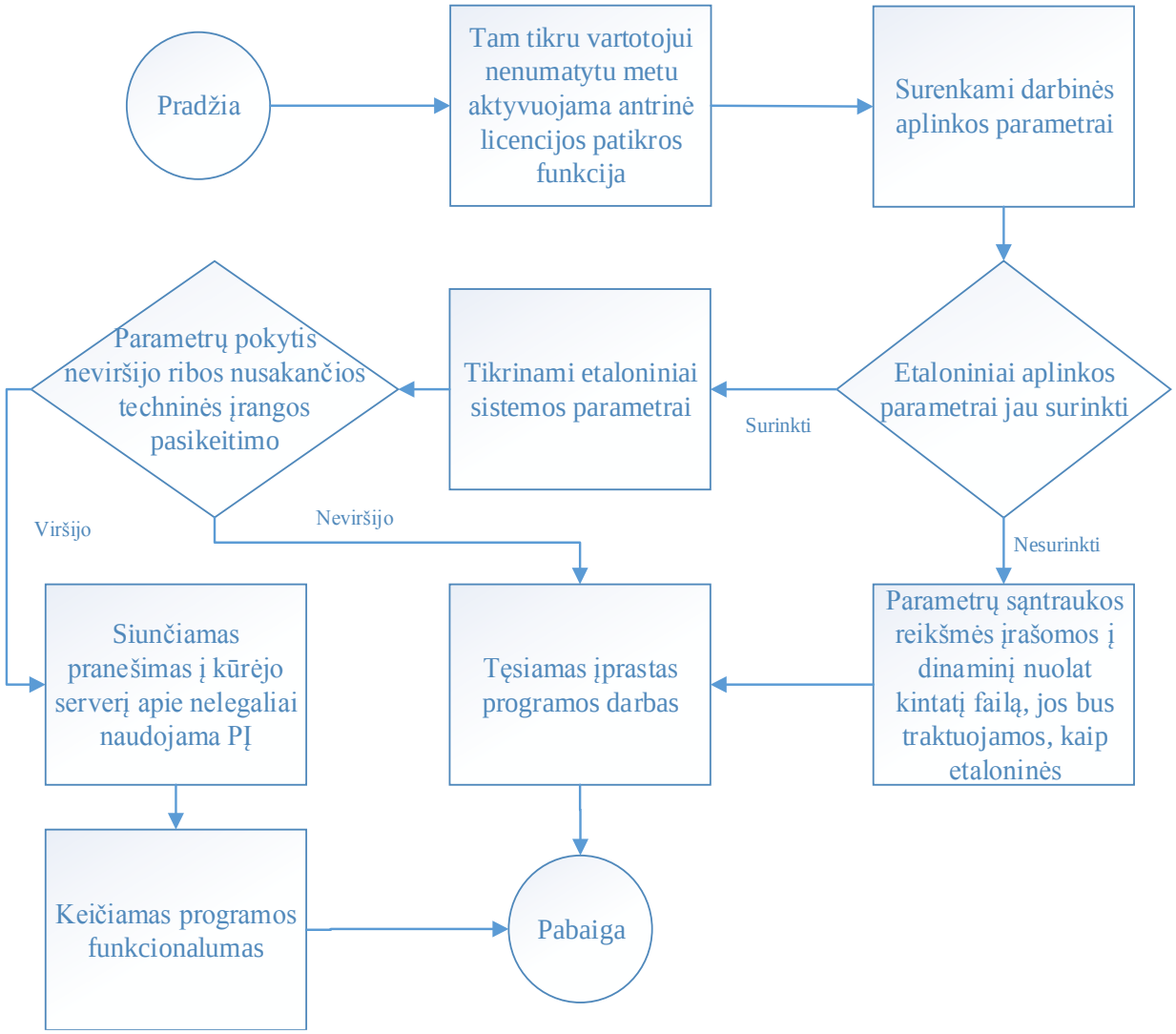
Antrinio licencijavimo procesas yra santykinai sudėtingesnis palyginus su pirminiu tiek savo struktūra, tiek nenumatytu ir aiškiai neapibrėžtu vykdymo laiku. Šio licencijavimo esminis aspektas yra vartotojui iš anksto nenumatomas vykdymo laikas ir ne iš karto pastebimas veikimo pakitimas, kurį pastebėti itin sunku. Pastebėjus nekorektiškai veikiančią PĮ, praktiškai taptų neįmanoma atstatyti jos į pradinę būseną. Atstatyti veikimą būtų galima tik tiksliai žinant, kuris failas pasikeitė ir kokios reikšmės ten turėtų būti.

Šis procesas sudarytas taip, kad aptiktų faktą, jog veikimo aplinka pasikeitė, tai yra programinė įranga buvo nukopijuota į kitą įrenginį. Tačiau licencijavimo modelis su nenumatytu laiku iškviečiamomis paslėptomis patikromis turi kelis nenuginčijamus trūkumus. Vienas iš jų – galimybė, kad antrinis licencijavimas niekada nebus iššaukiamas. Taip gali nutikti, jei paslėptas programinis kodas bus vykdomas tik itin retai vykdomoje programinės įrangos funkcijoje. Todėl reikia surasti funkcijas, kurios naudojamos vidutiniškai dažnai, tai yra ne kiekvieną kartą paleidžiant aplikaciją, bet ir ne itin retai. Antrasis šio licencijavimo trūkumas – jei programinės įrangos pirminė licencijos patikra bus apeita ir PĮ bus nukopijuota į kitą veikimo aplinką dar iki surenkant aplinkos parametrus antrinio licencijavimo moduliui. Šiuo atveju būtų surinkti naujos aplinkos parametrai kurie būtų traktuojami kaip etaloniniai.

## 2.6 Antrinio licencijavimo inicijavimas

Tam tikru vartotojui nenumatytu momentu aktyvuojama antrinės licencijos patikros funkcija. Vykdant ją surenkami darbinės aplinkos parametrai, kurie yra tam tikrų iš anksto numatytų parametrų, aptartų 2.3.1 poskyryje, rinkinys. Tuomet patikrinama ar aplinkos parametrai jau surinkti. Jei parametrai dar nebuvo surinkti, apskaičiuojamos jų santraukos reikšmės, surašomos į dinaminį failą ir tęsiamas įprastas programos darbas. Kitu atveju lyginamos etaloninių parametrų santraukos reikšmės su ką tik gautomis. Jei pasikeitė keletas darbinės aplinkos

parametrų lyginant su etaloniniais, PĮ funkcionalumas nėra keičiamas. PĮ funkcionalumas keičiamas tik jei pasiekiamas tam tikras iš anksto numatytas pasikeitusių parametrų kiekis atsižvelgiant į jų svarbą, kuri apibrėžta 3 lentelėje.




**7 pav.** Antrinės licencijos patikros proceso diagrama

Šio antrinio licencijavimo esminis privalumas, yra tas, kad jį sunku aptikti. Sunku numatyti, kad jis naudojamas programinėje įrangoje. Netgi numanant jo naudojimą, nelengva surasti kurios funkcijos metu jis iškviečiamas. Aptikus faktą, kad programinė įranga naudojama nelegaliai, vartotojas nėra apie tai įspėjamas ir, ko gero, gali iškarto nepastebėti, kad programa nefunkcionuoja taip kaip turėtų. Todėl netgi pastebėjus, kad įvyko tam tikri pokyčiai funkcionalume, nelegaliam vartotojui nebūtų aišku, kada šis pasikeitimas įvyko.

## 2.7 Programos funkcionalumo keitimas

Antrinio licencijavimo metu aptikus faktą, kad programinė įranga naudojamas neteisėtai, iškyla iššūkis, kaip pakeisti programos funkcionalumą. Šis funkcionalumo pakeitimas privalo

būti sunkiai aptinkamas programišiui, kuris siekia nulaužti programinę įrangą taip, kad jis galėtų būti naudojama su nelegaliu aktyvacijos kodu ar priminė licencijos patikra visiškai nebūtų taikoma.

Tarkime, kad programa atlieka tam tikrus skaičiavimus. Tokiu atveju, kiekvieną kartą būtų galima tam tikras skaičiavimams naudojamas konstantas užkrauti iš dinamiškai ir nuolat kintančio failo, o antrinio licencijavimo metu šiek tiek modifikuoti jas taip, kad kitą kartą atliekant skaičiavimus būtų gaunami neteisingi rezultatai, tačiau vartotojui nebūtų lengvai nuspėjamas faktas, kad tai netinkamos licencijos sukeltas reiškinys. Būtų atliekamas vienintelis failo turinio atnaujinimas, o kadangi šis failas yra dinaminis ir nuolat atnaujinamas, pats konstantų pakeitimo faktas taptų sunkiai aptinkamu. Taip pasikeitus konstantų failui naudojama programinė įranga veiktų nepakitusia seka, o gaunami rezultatai nebūtų visiškai tikslūs ir teisingi.

### 2.7.1 PĮ funkcionalumo keitimo teisinis reglamentavimas

Programinės įrangos funkcionalumo keitimas apie tai visiškai neinformavus vartotojo gali pažeisti Lietuvos Respublikos vartotojų teisių apsaugos [22] įstatymo antrąjį skirsnį. Tiksliau šio įstatymo 3 straipsnio (vartotojų teisės) 2 punktą – „įsigyti saugias, tinkamos kokybės prekes ar paslaugas“ ir 3 punktą – „gauti teisingą ir visapusišką informaciją valstybine kalba apie parduodamas prekes, teikiamas paslaugas“. Taip pat gali būti pažeistas Lietuvos Respublikos civilinio kodekso [23], šeštosios knygos – prievolių teisė 6.803 straipsnio – gamintojo (tiekėjo) teisės ir pareigos 2 punktas: „Gamintojas (tiekėjas), jeigu ko kita nenustato sutartis privalo, pirmą papunktį – parduoti tinkamos kokybės prekes ir garantuoti jų kokybę, parduoti prekes sutartyje nustatytais terminais ir mastu.“

Siekiant išvengti vartotojo teisių ar gamintojo pareigų įstatymų pažeidimo galutinio vartotojo licencijoje būtina nurodyti, kad gamintojas garantuoja tinkamą programinės įrangos veikimą tuo atveju, jei ji naudojama legaliai su galiojančia tam kompiuteriui skirta licencija.

## 2.8 Išvados

Siekiant suteikti programinei įrangai kuo didesnį apsaugos lygį, būtina sudaryti licencijavimo procesą, kuris būtų kiek įmanoma visapusiškai saugesnis. Jis patobulintų jau egzistuojančius licencijavimo metodus ir juos panaudotų programinės įrangos apsaugai.

Šio darbo metu bus naudojama dvigubo licencijavimo metodika. Šią metodiką sudaro dviejų lygių licencijavimas. Atviras vartotojui matomas priminis licencijavimas – kurio metu bus naudojamas aktyvacijos kodas siunčiamas tinklu į programinės įrangos kūrėjo serverius. Šio aktyvacijos kodo persiuntimui tinklu ir validacijai bus pasitelkiami kriptografiniai metodai. Antrinis licencijavimas bus paslėptas nuo vartotojo bei iškviečiamas nenumatytu metu. Jo metu bus renkami iš anksto numatyti darbinės aplinkos parametrai. Bus sekama ar pasikeitė reikšminga

šių parametrų dalis, kuomet būtų galima daryti nenugінčijamą išvadą, kad programinė įranga buvo nukopijuota į kitą kompiuterį apeinant pirminį licencijavimą. Aptikus šį faktą, vartotojas nėra apie tai informuojamas, tačiau programinė įranga pradeda veikti netiksliai ir neteisingai – jos funkcionalumas pakeičiamas.

Vartotojas iš anksto supažindinamas su tuo, kad programinės įrangos kūrėjas nėra atsakingas už tai, kad nelegaliai naudojama programinė įranga veiktų korektiškai. Šį teiginį jis patvirtina sutikdamas su galutinio vartotojo licencija.

# 3. PROGRAMINĖS ĮRANGOS APSAUGOS PROTOTIPO REALIZAVIMAS

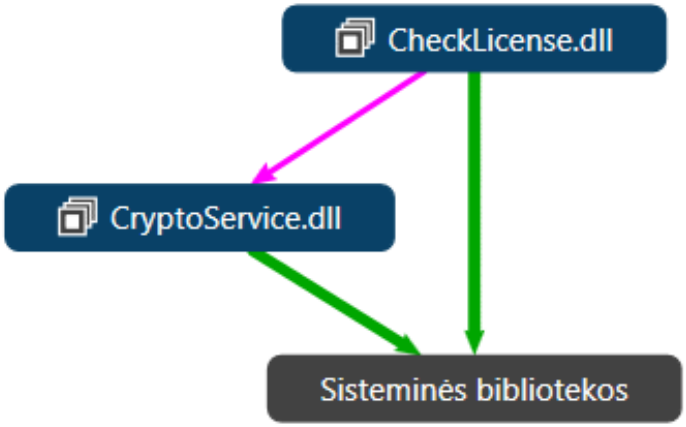
Šiame skyriuje bus apžvelgiamas realizuotas prototipas bei jo veikimo principas. Šį prototipą sudaro:

- Programinės įrangos kūrėjo serveryje veikianti žiniatinklio tarnyba (angl. *Web Service*).
- Apsaugota programinė įranga. Šiuo atveju bus naudojama apskaitos valdymo programa.

Taip pat bus naudojamos kelios papildomos bibliotekos, skirtos kriptografinių operacijų valdymui ar duomenų persiuntimui tinklu.

## 3.1 Žiniatinklio tarnybos veikimo principas

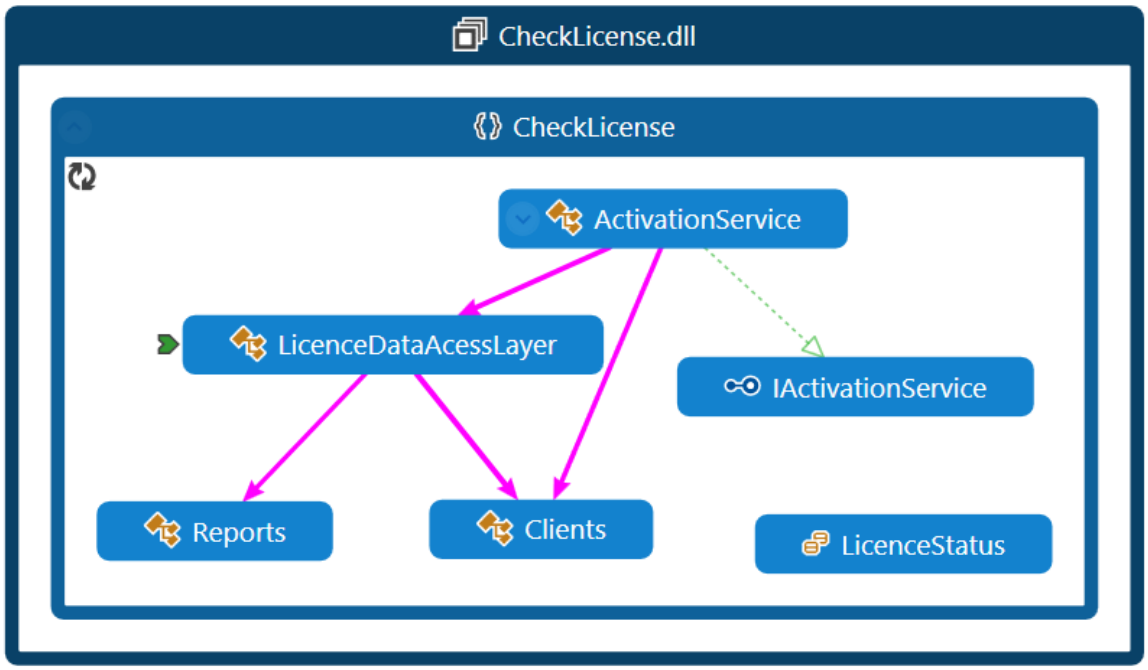
Žiniatinklio tarnyba skirta patikrinti aktyvacijos kodui. Ar jis jau nebuvo panaudotas bei ar jis išties yra teisingas. Šią tarnybą galima skirstyti į tris atskiras dalis – bibliotekas, kurių priklausomybių grafas pateiktas žemiau esančiame paveikslėlyje.




**8 pav.** Žiniatinklio tarnyboje naudojamų bibliotekų priklausomybių grafas

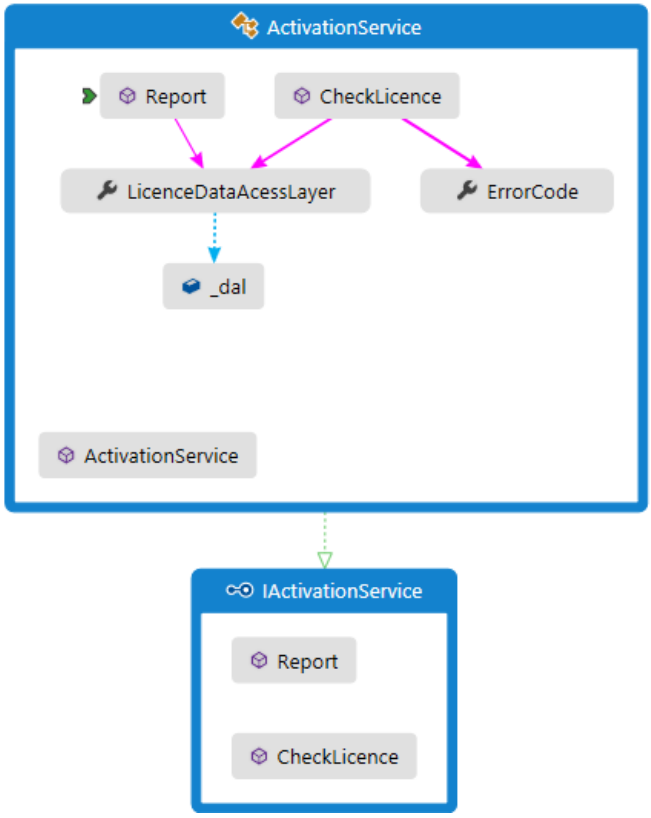
Iš pateikto grafo, galima matyti, kad kūrėjo serveryje veikiančią žiniatinklio tarnybą sudaro „CheckLicense“, „CryptoService“ bei kitos standartinės sisteminės bibliotekos. Pagrindinė šios žiniatinklio tarnybos biblioteka yra „CheckLicense“, kurios funkcija yra aktyvuoti programinę įrangą išduodant aktyvacijos atsaką bei priimti pranešimą apie aptiktą nelegalų programinės įrangos panaudojimą antrinio licencijavimo metu.

Detaliau panagrinėkime „CheckLicense“ biblioteką. Žemiau pateiktame paveiksle galima matyti šios bibliotekos objektų priklausomybių grafą. Kitos bibliotekos taip pat bus aptartos tolimesniuose šio skyriaus poskyriuose.




**9 pav.** „CheckLicense“ bibliotekos objektų priklausomybių grafas

Iš „CheckLicense“ bibliotekos objektų priklausomybių grafo, galima matyti, kad hierarchijos viršuje yra „ActivationService“, kuris aprašo „WCF“ (angl. *Windows Communication Foundation*) žiniatinklio tarnybos „IActivationService“ metodus. Žemiau pateikiamame objektų priklausomybės grafe galima matyti kokius metodus pateikia žiniatinklio tarnyba bei kokie objektai naudojami vykdant juos.




**10 pav.** Žiniatinklio tarnybą įgyvendinančių objektų grafas

Galima matyti, kad „IActivationService“ žiniatinklio tarnyba turi du metodus:

- „CheckLicense“ metodas yra skirtas patikrinti licencijos validumui.
- „Report“ metodas yra skirtas išsaugoti duomenis apie aptiktą nelegaliai veikiančią programinė įrangą. Tai gali būti įvairi informacija, šiuo atveju tiesiog išsaugomas kliento pavadinimas, su kurio licencija – nelegaliai paleista programinė įranga.

„CheckLicense“ metodo vykdomas išeities kodas pateikiamas žemiau esančiame paveiksle.

```
public string CheckLicence(string activationCode, string clientName)
{
    try
    {
        var client = LicenceDataAcessLayer.GetClient(clientName);
        if (client == null || client.ActivationStatus != LicenceStatus.Issued.ToString())
        {
            return ErrorCode;
        }

        using (var cryptoServiceProvider = new CryptoProvider())
        {
            cryptoServiceProvider.LoadAes(client.AesKey, client.AesIV);
            cryptoServiceProvider.LoadRsaProviderFromPrivateKey(client.PrivateRsaKey);
            var licence = cryptoServiceProvider.DecryptRsa(activationCode);
            if (client.LicenceKey != licence)
            {
                return ErrorCode;
            }

            var result = cryptoServiceProvider.EncryptAes(licence + cryptoServiceProvider.SignRsa(licence));
            LicenceDataAcessLayer.SetActivatedStatus(client);
            return result;
        }
    }
    catch
    {
        return ErrorCode;
    }
}
```

**11 pav.** „CheckLicence“ metodo metu vykdomas išeities kodas

Kaip matoma iš pateikto išeities kodo, į licencijos patikros metodą perduodami du parametrai. Tai yra aktyvacijos kodas ir kliento pavadinimas. Pradžioje yra patikrinama, ar toks klientas pirko programinę įrangą. Jei nepirko arba licencija išduota šiam klientui jau išnaudota, tai yra ji jau buvo aktyvuota arba jos statusas nėra išduota (angl. *Issued*) tiesiog grąžinamas klaidos kodas.

Esant teisingai nurodytam klientui ir neišnaudotai licencijai pradedama tikrinti atsiųstas aktyvacijos kodas. Šis kodas perduodamas užšifruotas viešuoju „RSA“ raktu, todėl jis pirmiausia iššifruojamas su klientu susietu privačiuoju „RSA“ raktu, kurį turi tik programinės įrangos kūrėjas. Tuomet sulyginami licencijos kodai, ar išduotas kodas parduodant programinę įrangą ir atsiųstas kodas sutampa. Jei kodai sutampa, grąžinama sėkmingas atsakas – „AES“ raktu užšifruota licencijos ir licencijos parašo kombinacija. Nesutapimo atveju – grąžinamas klaidos kodas.

Kitas prieinamas metodas, skirtas priimti duomenis apie nelegaliai naudojamą programinę įrangą yra „Report“. Šio metodo išeities kodas pateikiamas žemiau esančiame paveikslėlyje.

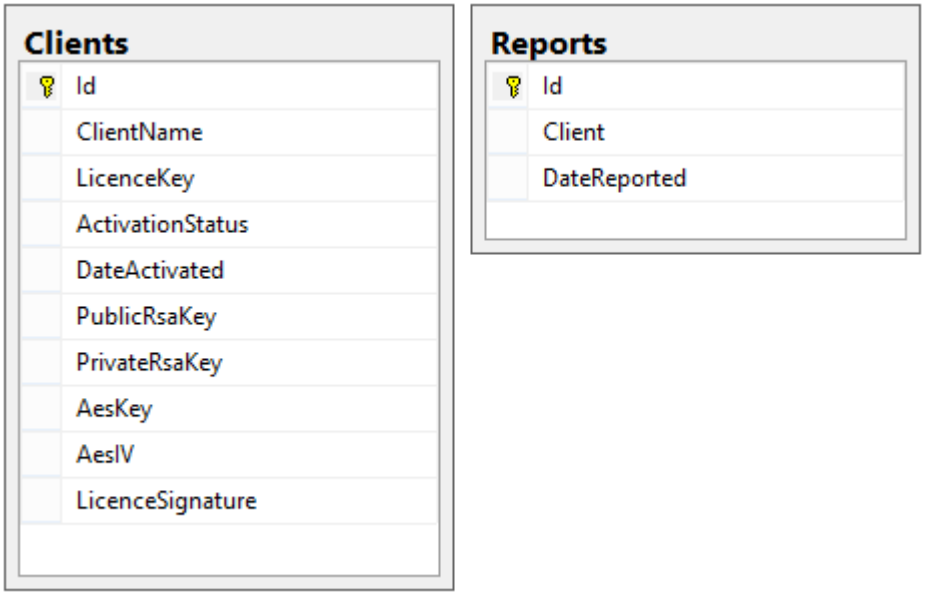
```
public void Report(string clientName)
{
    LicenceDataAcessLayer.AddReport(clientName);
}
```

**12 pav.** „Report“ metodo metu vykdomas išeities kodas

Pranešimo apie nelegaliai naudojamą programinę įrangą metodas yra kur kas paprastesnis nei licencijos patikros. Šis metodas tiesiog išsaugo kliento pavadinimą, kuris buvo aptiktas nelegaliai naudojantis programinę įrangą. Šią informaciją vėliau galima peržiūrėti ir imtis atitinkamų veiksmų.

### 3.1.1 Licencijavimo informacijos duomenų modelis

Licencijavimo duomenų modelis yra gana paprastas. Jį sudaro dvi duomenų lentelės – „Clients“ ir „Reports“. Šios lentelės pavaizduotos žemiau pateikiamame paveiksle.




**13 pav.** Licencijavimo informacijos duomenų modelis

„Clients“ lentelėje saugoma su klientu ir sugeneruota licencija susijusi informacija. Ši informacija nurodo klientą, jam išduotą aktyvacijos kodą, viešąjį ir privatųjį „RSA“ raktus, „AES“ raktą ir inicializacijos vektorių, pasirašytą aktyvacijos kodą bei datą kada buvo aktyvuota licencija. „Reports“ lentelėje saugoma informacija apie klientus, kurie buvo aptikti nelegaliai naudojantys programinę įrangą.

## 3.2 Vartotojo programoje veikiantis licencijavimo modulis

Diegiant programinę įranga surenkami aplinkos parametrai, paskaičiuojamos jų maišos funkcijų reikšmės ir išsaugomos faile. Diegimo metu surenkami šie parametrai:

- Procesoriaus architektūros tipas.
- Operatyviosios atminties dėtuvių kiekis.
- „BIOS“ serijos numeris.
- Pagrindinės plokštės identifikatorius.

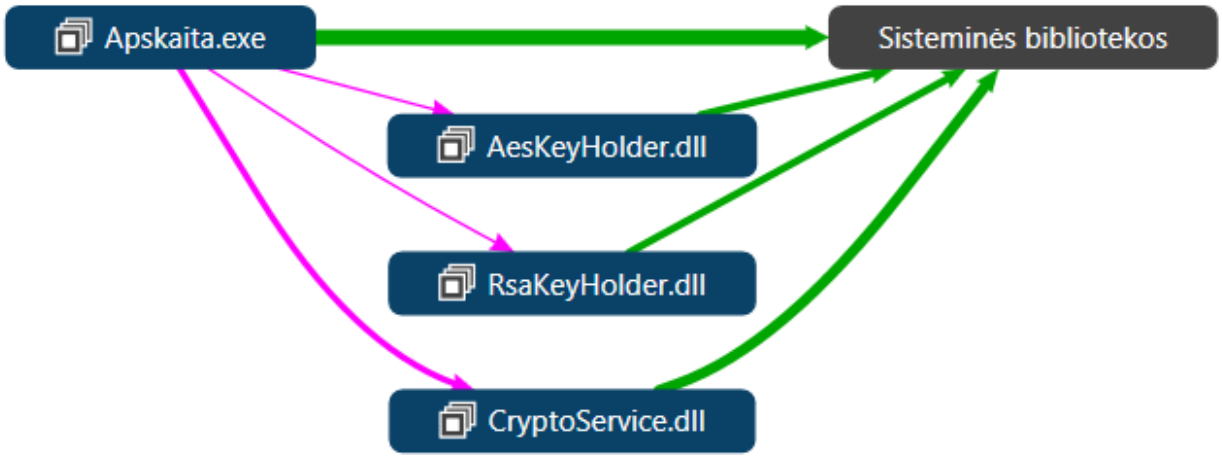
Kiekvieną kartą paleidus programinę įrangą, būtų tikrinami ar šie aplinkos parametrai nepasikeitė. Kadangi visi renkami parametrai turi aukštą svarbą, kuri apibrėžta 3 lentelėje, todėl bus traktuojama, kad kiekvienas parametras sudaro 25 proc. svarbą Darysime prielaidą, kad programinė įranga veikia kitame kompiuteryje kai suminė pasikeitusių parametrų svarbų reikšmė bus 75 proc. ar daugiau. Todėl programinės įrangos vykdymas bus stabdomas tik jei pasikeis 3 parametrai.

Antrinio licencijavimo metu bus surenkami kiti parametrai, nei diegimo metu. Bus surenkami šie parametrai:

- Tinklo plokštės „MAC“ adresas.
- Procesoriaus identifikatorius.
- Kompiuterio vardas.
- Sisteminio kietojo disko modelis.

Antrinio licencijavimo parametrai pasiskirstę taip, kad du iš jų turi aukštą svarba: tinklo plokštės „MAC“ adresas ir procesoriaus identifikatorius ir du vidutinę: kompiuterio vardas ir sisteminio kietojo disko modelis. Antrinio licencijavimo metu aukštos svarbos parametrai sudarys po 35 proc., o vidutinės – 15 proc. Pereinamoji riba, kada bus traktuojama, kad pasikeitė įrenginys kuriame veikia programinė įranga bus 60 proc. ar daugiau.

Žemiau pateiktame paveiksle vaizduojama apskaitos programinės įrangos bibliotekų priklausomybių grafas.




**14 pav.** Apskaitos programinės įrangos bibliotekų priklausomybių grafas

Iš pateikto bibliotekų priklausomybių grafų galima matyti, kad apskaitos programinė įranga naudoja „AesKeyHolder“, „RsaKeyHolder“, „CryptoService“ ir sistemines bibliotekas. Bibliotekos „AesKeyHolder“ ir „RsaKeyHolder“ skirtos laikyti kriptografiniams „AES“ ir „RSA“ raktams. Šie raktai įrašomi individualiai kiekvienam klientui. Žemiau pateiktame paveiksle matyti programinis kodas aprašantis „AES“ raktą ir inicializacijos vektorių sugeneruotą klientui ir išsaugota „AesKeyHolder“ bibliotekoje.

```
public static class Aes256
{
    public const string Key = "zwHdCQpYCHJ/cVK6RJ0Mzw==";
    public const string Iv = "zwHdCQpYCHJ/cVK6RJ0Mzw==";
}
```

**15 pav.** „AES“ kriptografinio rakto ir inicializacijos vektoriaus išeities kodas

Iš 15 pav. pateikto išeities kodo, galima matyti, kaip dinaminės bibliotekoj išsaugoma informacija skirta atlikti kriptografinėms operacijoms naudojant „AES“ kriptosistemą.

Žemiau pateikiamame paveiksle, galima matyti „RsaKeyHolder“ bibliotekoje saugoma viešąjį „RSA“ raktą.

```
public static class Rsa2048
{
    public const string PublicKey = "<RSAKeyValue>" +
                                    "<Modulus>" +
                                    "q+YKS6wQS6c9kJnE861/q8UJdZ890ZbISzx" +
                                    "EQJ/6S1WXkXJddDklInNmU7LIuRGEsL5lHKJvb1sbFvVdMAnjO/tT/6yx" +
                                    "uMtP0ycRjoM+KR7WaI7IKZOoLOZCyKgCOq5JJXeN3+EPOIQUespGZKyJN" +
                                    "qyfJgp7uGpsr5FKoGRij9VYXVAtyrx4ZyXBx+ri6USwW5A+7rgKtH/Uvw" +
                                    "vZ6O+/XcPB91xgPz5Bw5BPuhsFgoUIBRNqrmYqt0ipmiQw985jttWHG22" +
                                    "eUrGxgyCKQrP7RNXlG+ChhvP8XEH6cFhM6/it/S/RqtGWKeOCJl2MVfS+" +
                                    "UMceXhVnmGLSunYDv/9qfw==" +
                                    "</Modulus>" +
                                    "<Exponent>AQAB</Exponent>" +
                                    "</RSAKeyValue>";
}
```

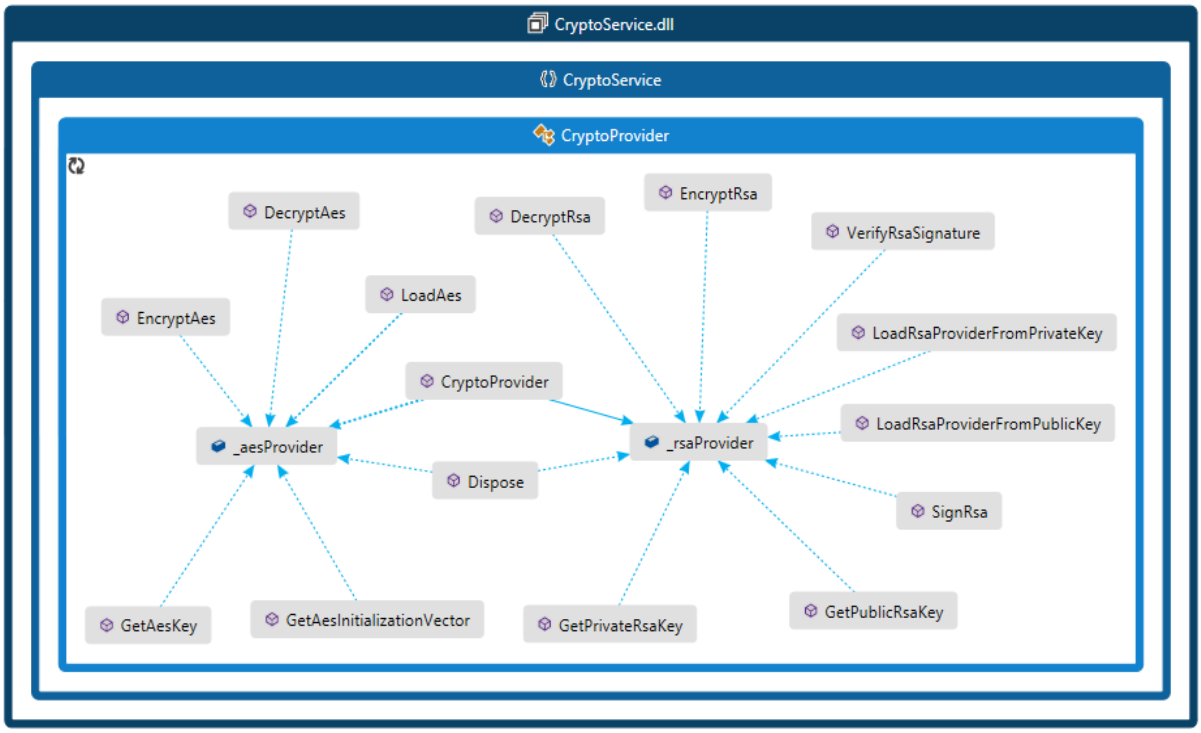
**16 pav.** „RSA“ kriptografinio rakto išeities kodas

Iš „RSA“ raktą aprašančio programinio kodo, galima matyti, kad klientui keliauja tik viešoji jo dalis. Ją sudaro modulis ir eksponentė.

Iš pateiktų 15 pav. ir 16 pav. galima matyti, kad „RSA“ rakto vien viešoji dalis yra kur kas sudėtingesnė nei „AES“.

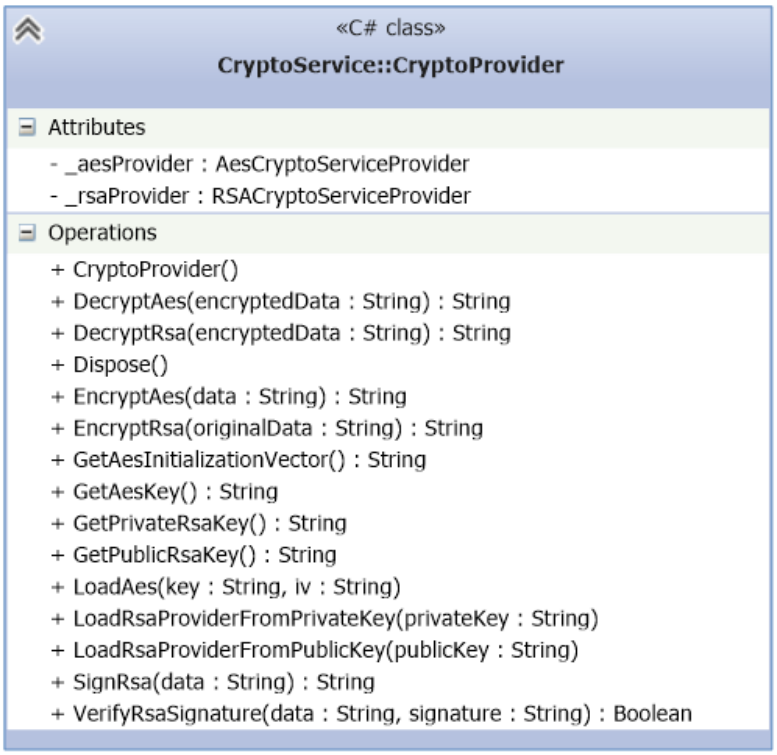
### 3.3 Kriptografinių metodų biblioteka

Kriptografinių metodų biblioteka skirta atlikti visoms šiame darbe minimoms kriptografinėms funkcijoms atlikti. Ji apima šifravimą ir iššifravimą naudojant „RSA“ bei „AES“ kriptosistemas, „RSA“ parašo generavimą ir jo validumo patikrinimą. Žemiau pateikiama metodų ir objektų priklausomybių grafas.




**17 pav.** „CryptoService“ bibliotekos metodų ir objektų priklausomybių grafas

Iš pateikto kriptografijos serviso bibliotekos grafo galima matyti, kad ši biblioteka apjungia ir prideda naują abstrakcijos lygmenį dviems sisteminėms bibliotekoms: „AesCryptoServiceProvider“ ir „RsaCryptoServiceProvider“ funkcionalumą. Detaliau išskirtas operacijas galima matyti „CryptoProvider“ klasės diagramoje, pateikiamoje žemiau esančiame paveiksle 18 pav. Jame galima matyti kokius bei kokio tipo parametrus priima ir grąžina operacijos.



**18 pav.** „CryptoProvider“ klasės diagrama

„CryptoProvider“ klasės pagrindinis uždavinys yra vykdyti visas su kriptografija susijusias operacijas. Šios operacijos:

- „EncryptAes“ ir „DecryptAes“ – užšifruoja ir iššifruoja duomenis naudojantis „AES“ algoritmu.
- „EncryptRsa“ ir „DecryptRsa“ – užšifruoja ir iššifruoja duomenis naudojantis „RSA“ algoritmu.
- „SignRsa“ ir „VerifyRsaSignature“ – atitinkamai sugeneruoja duomenų parašą ir patikrina sugeneruoto parašo validumą.
- „LoadRsaProviderFromPrivateKey“ ir „LoadRsaProviderFromPublicKey“ – paruošia kriptografinius raktus šifravimui, iššifravimui pateikiant privatų raktą bei tik užšifravimui, naudojant tik viešąjį raktą.
- „LoadAes“ – paruošia kriptografinius raktus šifravimui ir iššifravimui naudojantis „AES“ algoritmu.

Atidžiau panagrinėkime šifravimui skirtus metodus. Šių metodų išeities kodas pateikiamas žemiau esančiuose paveiksluose:

```
public string EncryptRsa(string originalData)
{
    var encryptedData = _rsaProvider.Encrypt(Encoding.ASCII.GetBytes(originalData), false);
    var encryptedStr = Convert.ToBase64String(encryptedData);
    return encryptedStr;
}
```

**19 pav.** Šifravimo „RSA“ algoritmu išeities kodas

Naudojant operaciją „EncryptRsa“ būtina perduoti originalius duomenis, kuriuos norima užšifruoti. Šifravimui naudojamas viešasis raktas perduotas operacijomis „LoadRsaProviderFromPrivateKey“ arba „LoadRsaProviderFromPublicKey“. Kviečiama operacija „Encrypt“ iš sisteminės klasės „RsaCryptoServiceProvider“, kuri užšifruoja duomenis. Tuomet gražintas rezultatas paverčiamas į tekstą užkoduotą „Base64“ koduote ir gražinamas rezultatas.

```
public string EncryptAes(string data)
{
    ICryptoTransform encryptor = _aesProvider.CreateEncryptor();

    using (MemoryStream msEncrypt = new MemoryStream())
    {
        using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
        {
            using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
            {

                swEncrypt.Write(data);
            }
            var encryptedData = msEncrypt.ToArray();
            return Convert.ToBase64String(encryptedData);
        }
    }
}
```

**20 pav.** Šifravimo „AES“ algoritmu išeities kodas

Naudojant operaciją „EncryptAes“ perduodami duomenys užšifruojami naudojant pateiktus: raktą ir inicializacijos vektorių. Sukuriamas atminties srautas bei objektas rašantis į šį srautą, kuris užšifruoja ir įrašo duomenis. Tuomet srautas konvertuojamas į tekstą užkoduotą „Base64“ koduote ir grąžinamas rezultatas.

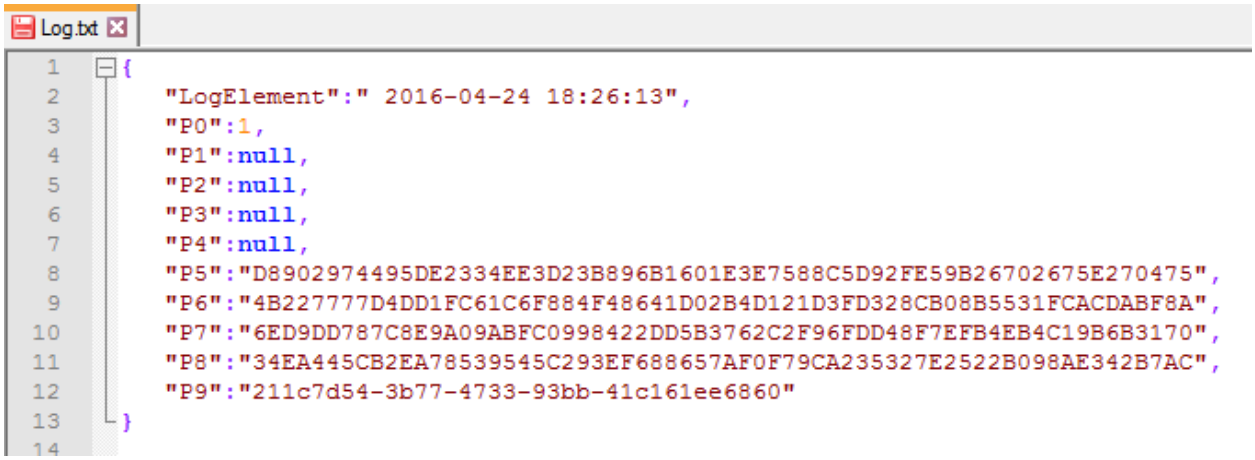
## 3.4 Išvados

Realizuotame prototipe išpildytos visos darbe numatytos apsaugos priemonės. Sukurta programinė įranga, kurią sudaro žiniatinklio tarnyba ir programinė įranga veikianti vartotojo kompiuteryje.

Vartotojo kompiuteryje veikianti programinė įranga apsaugota naudojant pirminę ir antrinę licencijos patikrą. Saugumui užtikrinti naudojama simetrinės ir asimetrinės kriptografijos metodai. Įgyvendintas antrinio licencijavimo funkcionalumas, kuris iškviečiamas vartotojui nenumatytu metu. Antrinis licencijavimas remiasi darbinės aplinkos parametrų rinkimu, saugojimu dinaminiame faile ir jų pokyčių sekimu. Aptikus reikšmingo parametrų kiekio pasikeitimą daroma išvada, kad programinė įranga veikia nelegaliai ir keičiamas jos funkcionalumas.

# 4. PROGRAMINĖS ĮRANGOS APSAUGOS BANDYMO REZULTATAI

Diegimo metu paruošiama pradiniai programinės įrangos veikimo failai, surenkami aplinkos parametrai. Surinktų parametrų maišos funkcijų reikšmės įrašomos į dinaminį failą. Šio failo pavyzdys pateiktas žemiau esančiame paveiksle 21 pav.




**21 pav.** Dinaminio failo turinys

Iš dinaminio failo turinio, nežinant, kas yra užkoduota po parametrais „P0“ – „P9“, jis nesuteikia jokios informacijos. Šio failo duomenys saugomi „JSON“ formatu. Kur parametrų reikšmės:

- „LogElement“ yra kintamoji duomenų dalis, nurodanti paskutinį laiką, kuomet buvo paleista programinė įranga.
- „P0“ – skaičiavimams naudojama konstanta.

„P1“ iki „P4“ – parametrai saugo aplinkos parametrų reikšmes naudojamas antriniam licencijavimui. Šie parametrai apibūdina:

- „P1“ – Tinklo plokštės „MAC“ adresas.
- „P2“ – Procesoriaus identifikatorius.
- „P3“ – Kompiuterio vardas.
- „P4“ – Sisteminio kietojo disko modelis

Parametrai nuo „P5“ iki „P8“ saugo aplinkos parametrų reikšmes naudojamas pirminiam licencijavimui, Šie parametrai apibūdina:

- „P5“ – Procesoriaus architektūros tipas.
- „P6“ – Operatyviosios atminties dėtuvių kiekis.
- „P7“ – „BIOS“ serijos numeris.
- „P8“ – Pagrindinės plokštės identifikatorius.

Pav. 21 pateiktame paveiksle parametrų reikšmės „P1“ – „P4“ yra „null“, kadangi dar nebuvo aktyvuota antrinio licencijavimo funkcija.

## 4.1 Programinės įrangos paleidimo lėtinimas

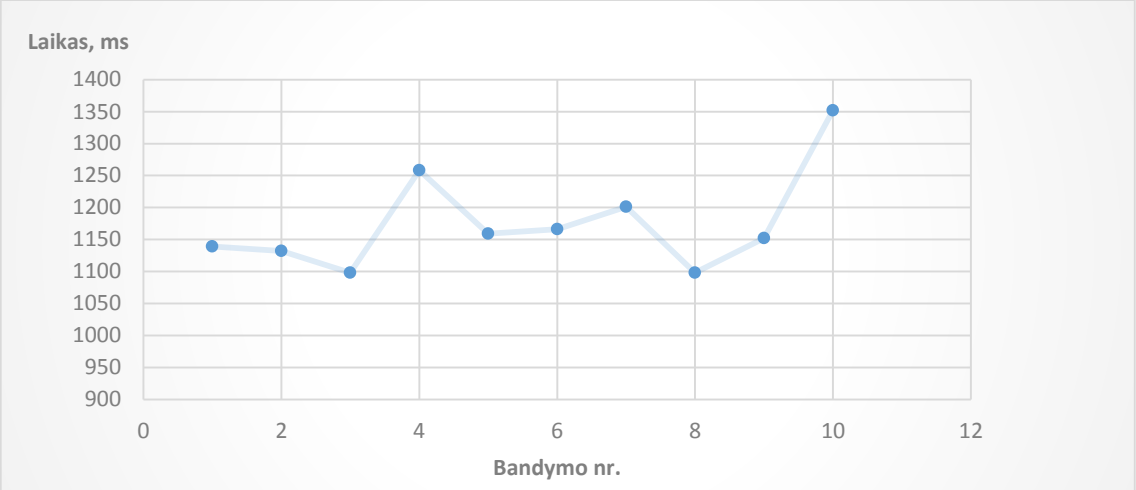
Dėl atsiradusių papildomų žingsnių, kurie vykdomi paleidus programinę įrangą, pailgėja laikas, per kurį ji galutinai užkraunama, panagrinėkime, kokį laiko tarpą sudaro licencijos patikros procesas: registro įrašo patikrinimas naudojant kriptografinius metodus, parametrų surinkimas ir sulyginimas su etaloniniais.

Bandymo metu, atskirai išmatuoti laiko tarpai per kuriuos:

- Surenkami darbinės aplinkos parametrai, apskaičiuojamos jų maišos reikšmės ir palyginamos su reikšmėmis išsaugotomis dinaminiame faile.
- Iš „Windows“ registro nuskaitoma užšifruota licencija, užkraunami kriptografiniai raktai ir paruošiama kriptografijos biblioteka, patikrinama ar licencijos parašas yra teisingas, naudojant viešąjį „RSA“ raktą.

Bandymo metu gauti laikai gali skirtis naudojant skirtingus kompiuterius.

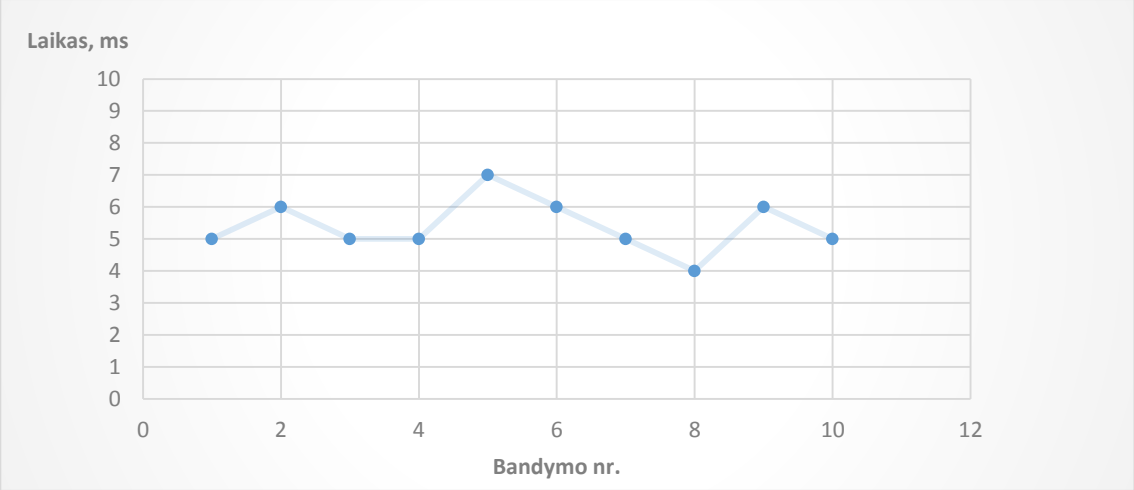
Iš 22 pav. galima matyti, kad darbinės aplinkos parametrų surinkimas vidutiniškai trunka 1175,5 ms.




**22 pav.** Laikas per kurį surenkami darbinės aplinkos parametrai PĮ paleidimo metu

Taip pat minimalus ir maksimalus laikas skiriasi 254 ms., kuris svyruoja atitinkamai nuo tuo metu esančios sistemos apkrovos.

Taip pat buvo išmatuotas laikas skirtas patikrinti „Windows“ registro įrašui. Bandymo rezultatai pateikiami 23 pav. Iš šio paveikslo galima matyti, kad registro duomenų patikrinimas vidutiniškai trunka 5.4ms, tai yra apytiksliai 200 kartų greičiau, nei aplinkos parametrų patikrinimas. Todėl galutiniam programos paleidimo laikui jis įtakos beveik nesudaro.




**23 pav.** Laikas per kurį patikrinamas įrašas „Windows“ registre PĮ paleidimo metu

Atlikus greitaveikos bandymą, nustatyta, kad galutinis programinės įrangos paleidimo laikas vidutiniškai padidės 1180,9ms, iš kurių 1175,5ms bus tikrinami aplinkos parametrai ir 5,4ms „Windows“ registro įrašas.

## 4.2 „Windows“ operacinės sistemos registro įrašas

Sėkmingai aktyvavus programinę įrangą, gautas atsakymas iš kūrėjo serverių įrašomas į „Windows“ registrą. Šis įrašas sudarytas taip:

- Pasirašomas aktyvacijos kodas.
- Pasirašytas aktyvacijos kodas ir sugeneruotas parašas sujungiami ir užšifruojami „AES“ kriptografiniu raktu.

Atsakas įrašomas į atskirai programai skirtą registro įrašą, kurio turinys pavaizduotas 24 pav. paveiksle.

| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| LK | REG_SZ | ME5mAQ7eSE1ll+kPrpbqmRTltYntdDDhF2gzl6Z6+stwFWKnJ3g8Ppc0CLZU2UQOnFJR3WrSHls1QC7DJrEiutM8GcLGXeLMsMKxJ3LD1qN5... |

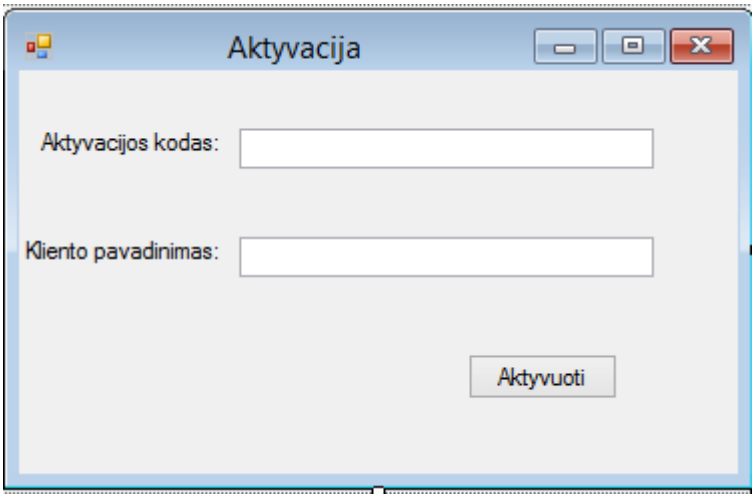
**24 pav.** „Windows“ registro įrašo turinys

Iš šio paveikslo galima matyti, kad registro įrašo LK duomenys yra visiškai neperskaitomi žmogui. Programinė įranga, neradusi šio registro įrašo ar radusi neteisingą, paleidimo metu, prašytų vartotojo aktyvuoti programinę įrangą. Šį įrašą praktiškai neįmanoma sufalsifikuoti, kadangi jis turi būti pasirašytas „RSA“ raktu, kurį turi tik programinės įrangos kūrėjas.

## 4.3 Programos paleidimas nukopijavus

Nukopijavus programinę įrangą, į kitą kompiuterį ji nepasileis, kadangi nesutaps diegimo metu surinkti parametrai su darbinės aplinkos parametrais. Tačiau emuliuojant šiuos parametrus, kadangi buvo galima aptikti, kad jie rinkti diegimo metu, programinę įrangą pavyks paleisti.

Tačiau paleista programinė įranga prašys, suvesti kliento pavadinimą ir aktyvacijos kodą kaip pavaizduota **Error! Reference source not found.**. paveiksle, kadangi nebus rastas registro įrašas, uris nusakytų, kad programinė įranga jau aktyvuota.




**25 pav.** Programinės įrangos aktyvacijos langas

Programišiams nukopijavus ir operacinės sistemos „Windows“ registro reikšmę, programinė įranga pasileistų, būtų sėkmingai nulaužtas pirminis licencijos patikros procesas ir programine įranga veiktų nelegaliai. Tačiau aktyvavus antrinės licencijos patikros procesą būtų tikrinami parametrai surinkti praeitame kompiuteryje ir surinkti šiame kompiuteryje. Pirminiame kompiuteryje surinktų parametrų sąntraukos reikšmės pavaizduotos 26 pav. paveiksle. Kurių nešifruotos reikšmės yra:

- „P1“ – Tinklo plokštės „MAC“ adresas: pirminė reikšmė: "AC220B1AEFC9"
- „P2“ – Procesoriaus identifikatorius: pirminė reikšmė: "BFEBFBFF000306A9"
- „P3“ – Kompiuterio vardas: pirminė reikšmė: "POVILO"
- „P4“ – Sisteminio kietojo disko modelis: pirminė reikšmė: "F6E38541B2B0115B"


```
    "P0": 1,
    "P1": "505AEEA31E3F0391613D7B6C1C3B134CFA659CFB92CB706BD9387894238O7EFC",
    "P2": "5F944592C74518D249BB3E5D58B884C8A2BCCA5C7B58CC91E7A64B634EBFC015",
    "P3": "3DEA127C2392DA34B22149362DA5965C96DE38F3AB29CE3FFF3396151CAB132F",
    "P4": "1A310B71C5B2E5FFC0D047A21EFA238550B8BF72174E007DB5CB1D1EFAB8A5C2",
```


**26 pav.** Originalios antrinio licencijavimo aplinkos parametrų reikšmės

Sėkmingai imituojant kompiuterio aplinkos parametrus surinktus priminio licencijavimo metu ir pridėjus registro įrašą, kuriame yra sėkmingos aktyvacijos kodas buvo paleista programinė įranga, kurios parametrai atitinkamai yra:

- „P1“ – "333DDCF474BA"
- „P2“ – " BFEBFBFF0001067A"
- „P3“ – "DOVIL"
- „P4“ – "A8B46584B2B0105C"

Kadangi visi licencijos patikros metu tikrinami parametrai nesutampa, programinė įranga traktavo, kad pasikeitė vykdymo aplinka tai atsispindi dinaminiame faile, kuriame „P0“ parametro reikšmė pasikeitė iš „1“ į „1.2“. Galutinis dinaminio failo turinys pavaizduotas 27 pav. čia galima matyti, kad visi kiti parametrai išliko nepakitę išskyrus „P0“ parametrą . Kadangi šis parametras buvo naudojamas įvairiems skaičiavimams, tai ir patys programinėje įrangoje atliekami skaičiavimai tapo netikslūs ir neteisingi.

Log.txt


```
{
    "LogElement": " 2016-05-08 16:15:23",
    "P0": 1.2,
    "P1": "505AEEA31E3F0391613D7B6C1C3B134CFA659CFB92CB706BD938789423807EFC",
    "P2": "5F944592C74518D249BB3E5D58B884C8A2BCCA5C7B58CC91E7A64B634EBFC015",
    "P3": "3DEA127C2392DA34B22149362DA5965C96DE38F3AB29CE3FFF3396151CAB132F",
    "P4": "1A310B71C5B2E5FFC0D047A21EFA238550B8BF72174E007DB5CB1D1EFAB8A5C2",
    "P5": "D8902974495DE2334EE3D23B896B1601E3E7588C5D92FE59B26702675E270475",
    "P6": "4B227777D4DD1FC61C6F884F48641D02B4D121D3FD328CB08B5531FCACDABF8A",
    "P7": "6ED9DD787C8E9A09ABFC0998422DD5B3762C2F96FDD48F7EFB4EB4C19B6B3170",
    "P8": "34EA445CB2EA78539545C293EF688657AF0F79CA235327E2522B098AE342B7AC",
    "P9": "211c7d54-3b77-4733-93bb-41c161ee6860"
```


**27 pav.** Galutinis dinaminio failo turinys po aptikto nelegalaus panaudojimo fakto

## 4.4 Išvados

Bandymo metu buvo išmatuota greitaveikos pakitimas paleidžiant programinė įrangą. Šis pakitimas įtakojamas dviejų faktorių:

- Aplinkos parametrų surinkimo.
- Operacinės sistemos „Windows“ registro įrašo nuskaitymo ir validavimo.

Nustatyta, kad šie faktoriai gana minimaliai įtakoja paleidimo laiką testuotame kompiuteryje, kur vidutinis paleidimo vėlinimas sudaro apie 1180,9 ms.

Išbandytas licencijavimo modulio patikimumas nukopijuojant programą į kitą kompiuterį. Programinė įranga, kaip ir tikėtasi, neveikė vos ją nukopijavus. Kai buvo suimituoti originalaus kompiuterio aplinkos parametrai renkami diegimo metu ir sukurtas teisingas įrašas registre, tuomet pradėjo veikti ir pati programa.

Nukopijuotoje ir veikiančioje programinėje įrangoje vartotojui aktyvavus paslėptąjį licencijavimą, buvo aptiktas faktas, kad pasikeitė vykdymo aplinka. Tuomet pakeista parametro saugomo dinaminiame faile reikšmė, kuri įeina į įvairias skaičiavimo formules. Taip pasikeitė programinės įrangos funkcionalumas, ji pradėjo veikti neteisingai ir netiksliai.

# 5. IŠVADOS

Nelegaliai naudojama programinė įranga yra svarbi problema Lietuvos ir viso pasaulio kontekste. Remiantis statistiniais duomenimis, nelegalios ar nulaužtos programinės įrangos naudojimas sudaro virš 40 proc.

Nors programinės įrangos kūrėjai vis daugiau investuoja į jų programinės įrangos apsaugą, tačiau dar nėra sukurtas visiškai saugus sprendimas, užtikrinantis visapusę apsaugą nuo nelegalaus naudojimo. Siekiant išspręsti šią problemą, šio darbo metu buvo pasiūlytas patobulintas licencijavimo metodas.

Pasiūlytas metodas apima dviejų lygių licencijavimą – pirminį, matomą vartotojui, ir antrinį, paslėptą nuo vartotojo. Pirminį licencijavimą sudaro programinės įrangos aktyvacija internetu ir veikimo aplinkos parametrų sekimas paleidžiant programinę įrangą, jam aptikus nelegalumą, stabdomas programinės įrangos veikimas. Antrinį licencijavimą sudaro atsitiktiniu, vartotojui nežinomu metu aktyvuota funkcija, kuri seka skirtingų aplinkos parametrų reikšmes ir aptikus faktą, kad pasikeitė vykdymo aplinka, pakeičia programos funkcionalumą.

Sukurtas patobulintas licencijavimo prototipas veikia kaip numatyta. Šio metodo ir prototipo veikimas pagrįstas vartotojo nenumanymu apie antrinį licencijavimą. Bandymo metu išmėginus nukopijuotą programinę įrangą ir nulaužus pirminį licencijavimą, kaip ir tikėtasi, programinės įrangos funkcionalumas pasikeitė.

# LITERATŪRA


1. Lietuvos Respublikos ryšių reguliavimo tarnybos tinklalapis. [žiūrėta 2014-12-15]. Prieiga per internetą: http://www.esaugumas.lt/lt/saugus-darbas-internete/piratavimas.html
2. CONNER, K. R.; RUMELT, R. P. Software Piracy: An Analysis of Protection Strategies. [žiūrėta 2014-12-15]. Prieiga per internetą: http://pubsonline.informs.org/doi/pdf/10.1287/mnsc.37.2.125
3. The Software Alliance tinklalapis. [žiūrėta 2014-12-15]. Prieiga per internetą: http://ww2.bsa.org/country.aspx?sc_lang=lt
4. ZITTRAIN, J. L. *The Future Of The Internet And How To Stop It*. 2009, p. 30 – 80.
5. MADOU, M.; ANCKAERT,B.; SUTTER, B. D.; BOSSCHERE, K. D. *Hybrid Static-Dynamic Attacks against Software Protection Mechanisms.* 2005, p. 1 – 3 [žiūrėta 2016-04-24] Prieiga per internetą: http://users.elis.ugent.be/~brdsutte/research/publications/2005DRMmadou.pdf
6. KAZANAVIČIUS, E; TOLDINAS, J; ČEPONIS, J. *Programų sauga*, Kaunas, 2011, p. 56 – 169 .
7. WEOBLEWSKI, G. *General Method of Program Code Obfuscation.* [žiūrėta 2015-16-14]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.9052&rep=rep1&type=pdf&mode=1
8. NAGRA, J.; COLLBERG, C. *Surreptitious Software – Obfuscation, Watermarking, and Tamperproofing for Software Protection*. 2009, p. 16 – 35.
9. „Red Gate Reflector“ tinklalapis [žiūrėta 2015-11-15]. Prieiga internete: http://www.red-gate.com/products/dotnet-development/reflector/
10. „Jet Brains“ tinklalapis [žiūrėta 2015-12-16]. Prieiga per internetą: https://www.jetbrains.com/decompiler/.
11. „Pnf Software“ tinklalapis, *Reverse Engineering for Professionals* [žiūrėta 2015-12-16]. Prieiga per internetą: https://www.pnfsoftware.com/.
12. „Babel4Net“ tinklalapis [žiūrėta 2016-04-09]. Prieiga per internetą: http://www.babelfor.net/.
13. CHIKOFSKY, E.; CROSS, J. *Reverse Engineering and Design Recovery: A Taxonomy*. 1990, p. 7 – 20.
14. SYSTA, T. *Static and Dynamic Reverse Engineering Techniques for Java Software Systems*. 2000, p. 10 – 20.
15. TONELLA, P.; POTRICH, A. *Reverse Engineering of Object Oriented Code*. 2005, p. 15 – 20.
16. SAKALAUSKAS, E.; LISTOPADSKIS, N.; DOSINAS, G. S. *Kriptografijos teorija*. 2008, p. 7 – 14.
17. ELMINAAM, D. S. A.; KADER, H. M. A.; HADHOUD, M. M. *Evaluating The Performance of Symmetric Encryption Algorithms*. 2010. [žiūrėta 2016-04-10] Prieiga per internetą: http://isrc.ccs.asia.edu.tw/ijns/contents/ijns-v10-n3/ijns-2010-v10-n3-p213-219.pdf
18. DELFS, H.; KNEBL,H. *Introduction to Cryptography. Principles and Applications. Second Edition*. 2008, p. 1 – 77.
19. BARKER, E; JOHNSON, D.; SMID, M. *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*. 2007 [žiūrėta 2016-04-10]. Prieiga per internetą: http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf.
20. KUMAR, A. *A Palmprint Based Cryptosystem using Double Encryption*. p. 4 – 5. [žiūrėta 2016-04-10]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.4015&rep=rep1&type=pdf

21. BARKER, E.; ROGINSKY, A. *Transitions: Recommendation for Transitioning the Use of Cryptographiv Algorithms and Key Lengths.* NIST Special Publication 800-131A, 2011. [žiūrėta 2016-04-10]. Prieiga per internetą: http://www.gocs.eu/pages/fachberichte/archiv/075-sp800-131A.pdf
22. Lietuvos Respublikos Seimo tinklalapis, *Lietuvos Respublikos Vartotojų Teisių Apsaugos Įstatymas* [žiūrėta 2016-01-26]. Prieiga per internetą: http://www3.lrs.lt/pls/inter2/dokpaieska.showdoc_l?p_id=291694
23. „Infolex“ tinklalapis*, Lietuvos Respublikos civilinis kodeksas. Šeštoji knyga. Prievolių teisė 6.803 straipsnis* [žiūrėta 2016-01-26]. Prieiga per internetą: http://www.infolex.lt/ta/12755:str6.803